Containerizing Distributed Pipes

Hagen Tönnies
www.linkedin.com/in/hagen-toennies

Special Thanks to @CQnib

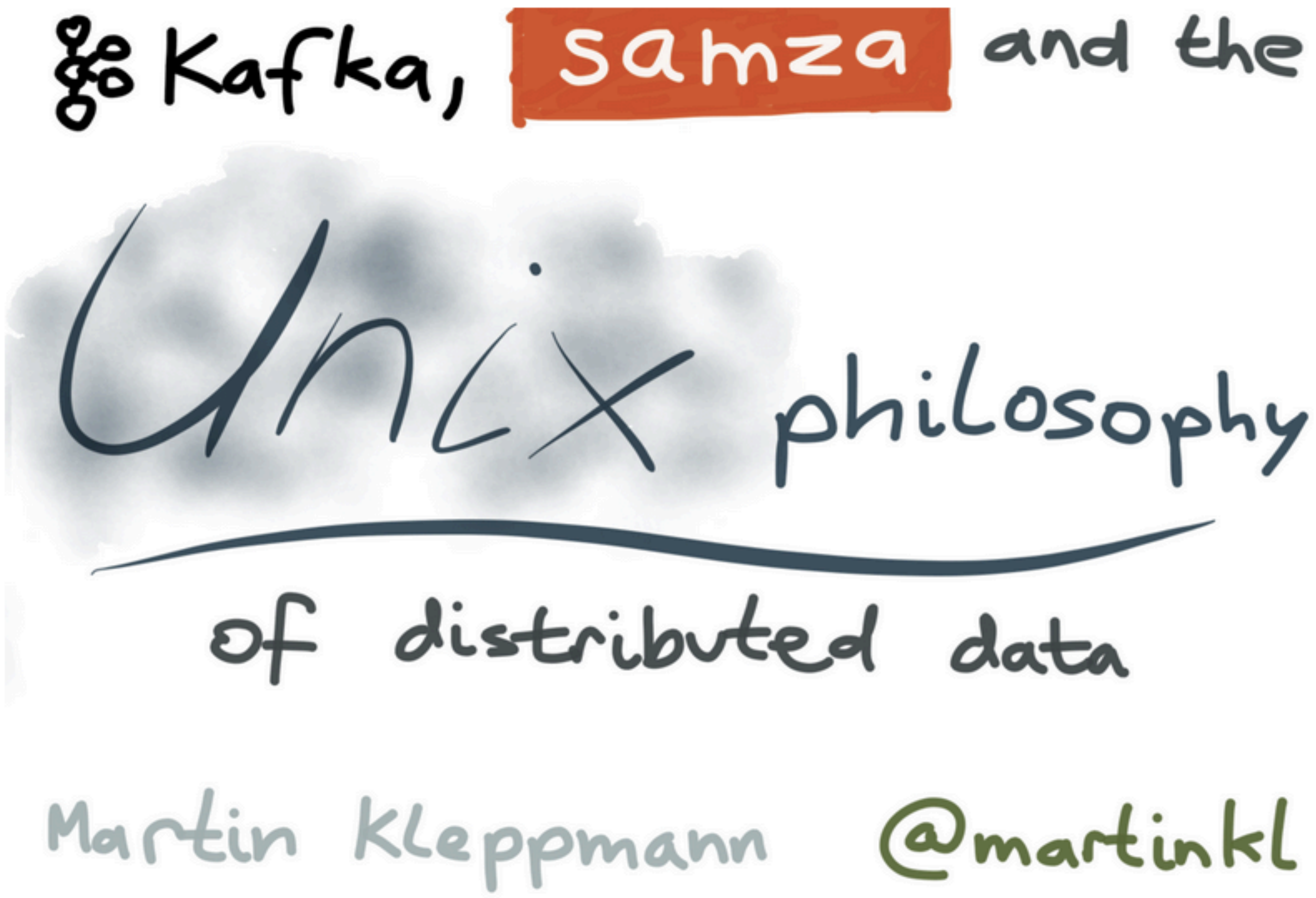
Agenda

- **Background**
- **Distributed Tools**
- **Containerizing**
- **Recap**

Background

Buzzwords

Background


Kafka, samza and the
Unix philosophy
of distributed data
Martin Kleppmann
@martinkl

PIPES

Unix Pipe Recap

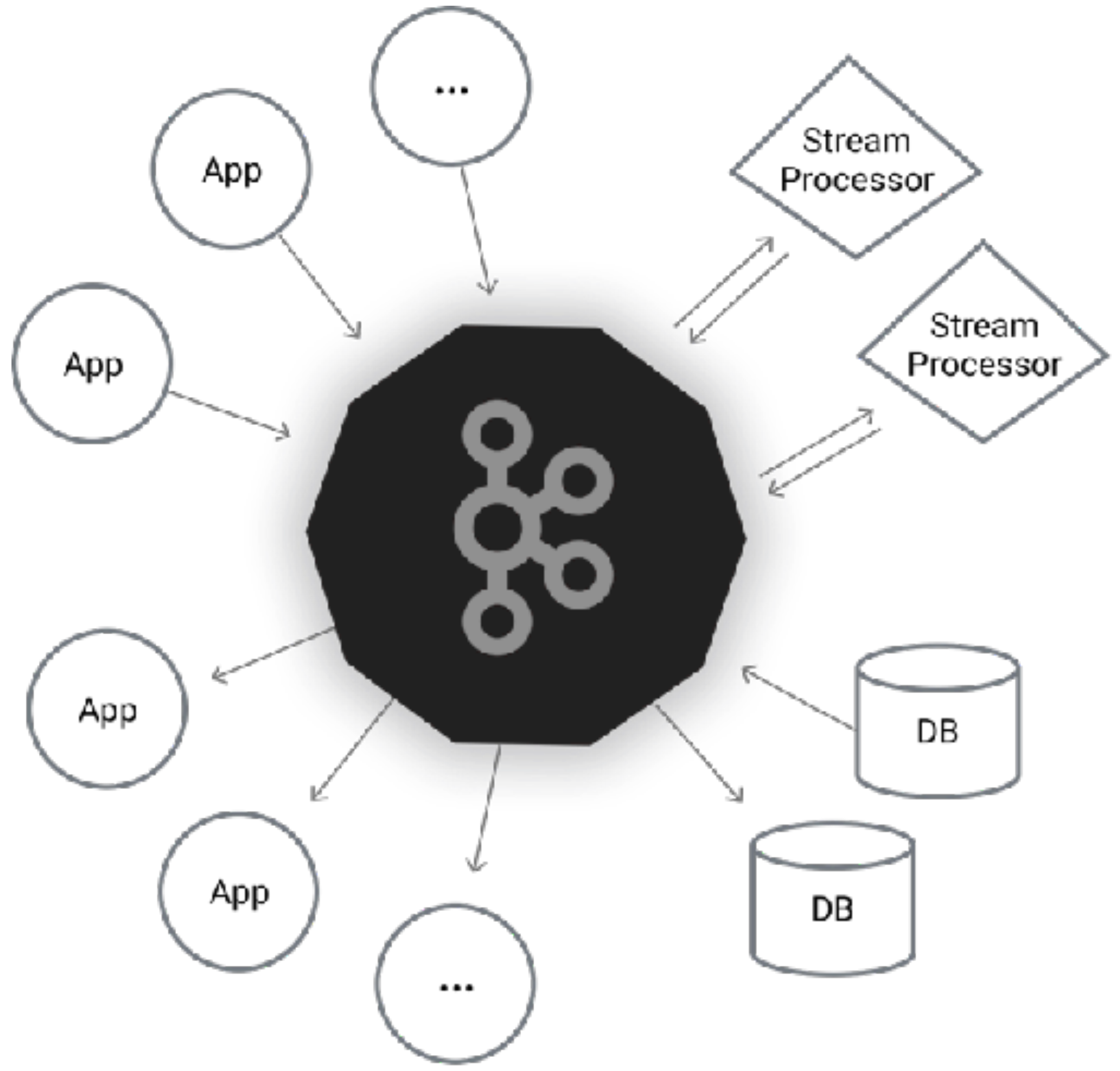
- […]In Unix-like computer operating systems, **a pipeline** is a **sequence** of **processes chained together** by their standard streams, so that the **output** of each **process** (stdout) feeds directly as **input** (stdin) **to the next one**.

Unix Philosophy

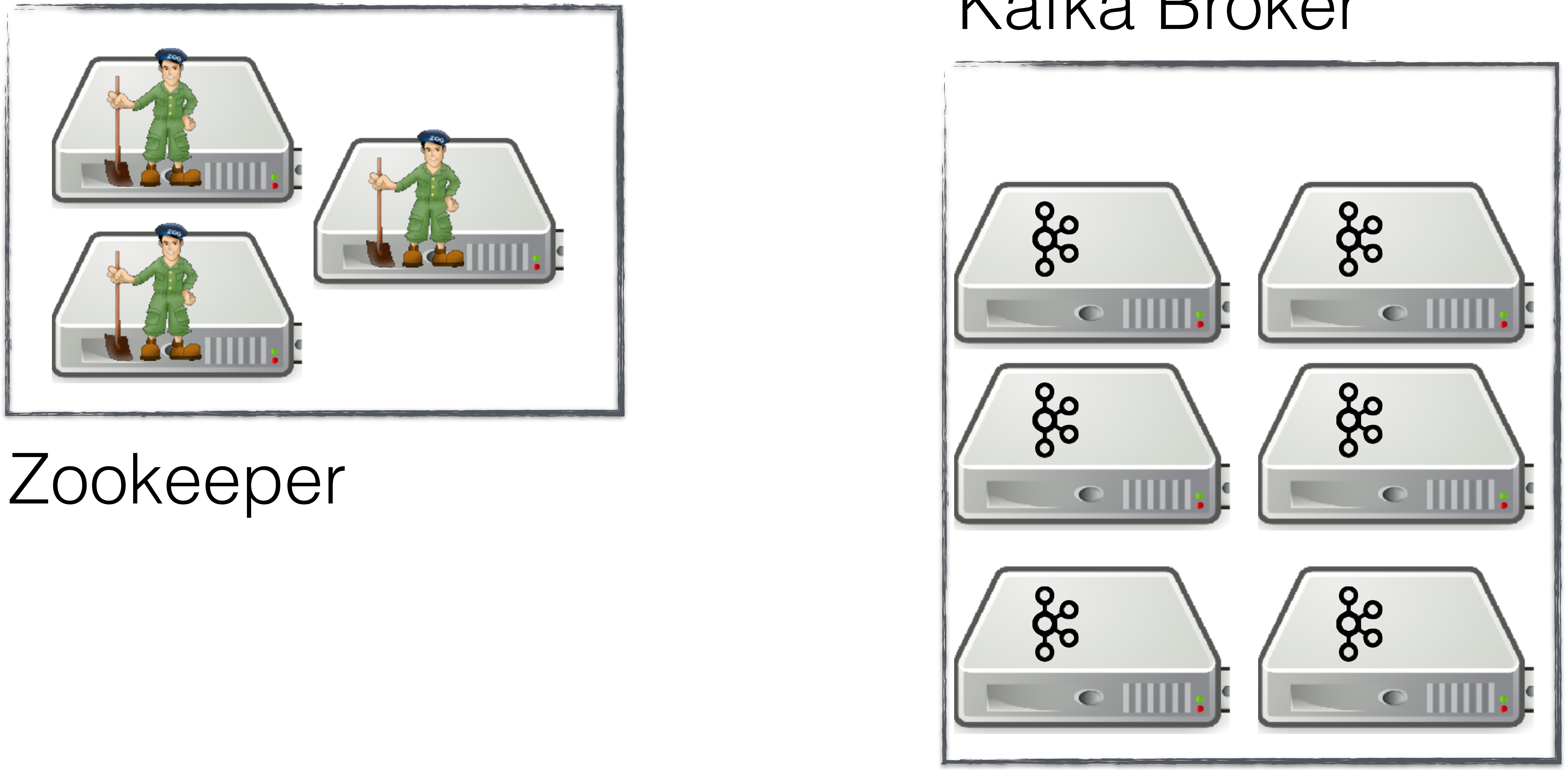
- Write **programs** that **do one thing** and **do it well**.
- Write **programs** to **work together**.
- Write **programs to handle text streams**, because that **is a universal interface**.

KAFKA

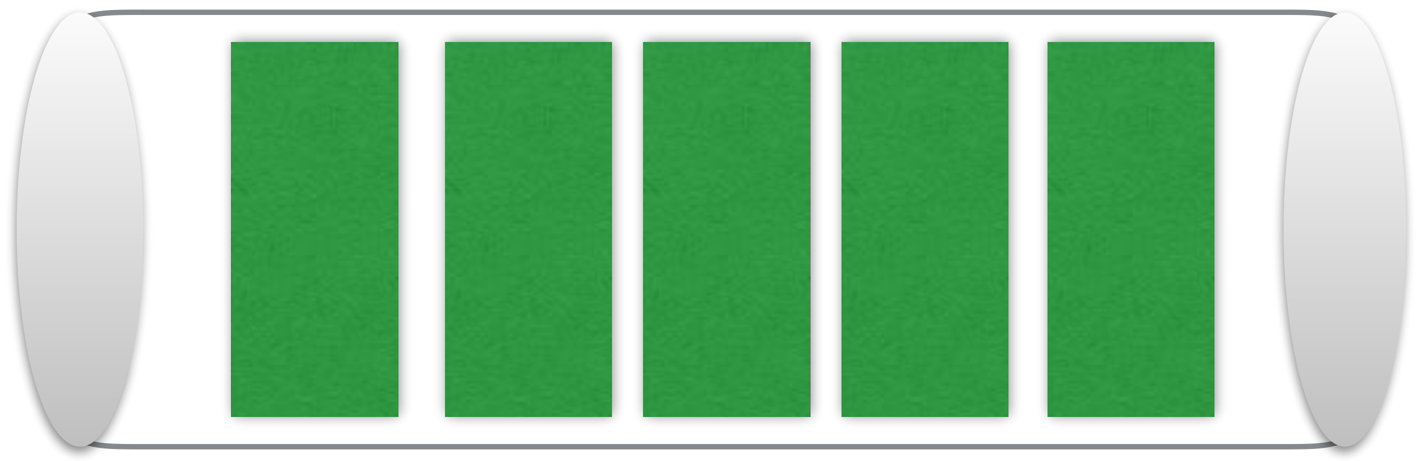
Background



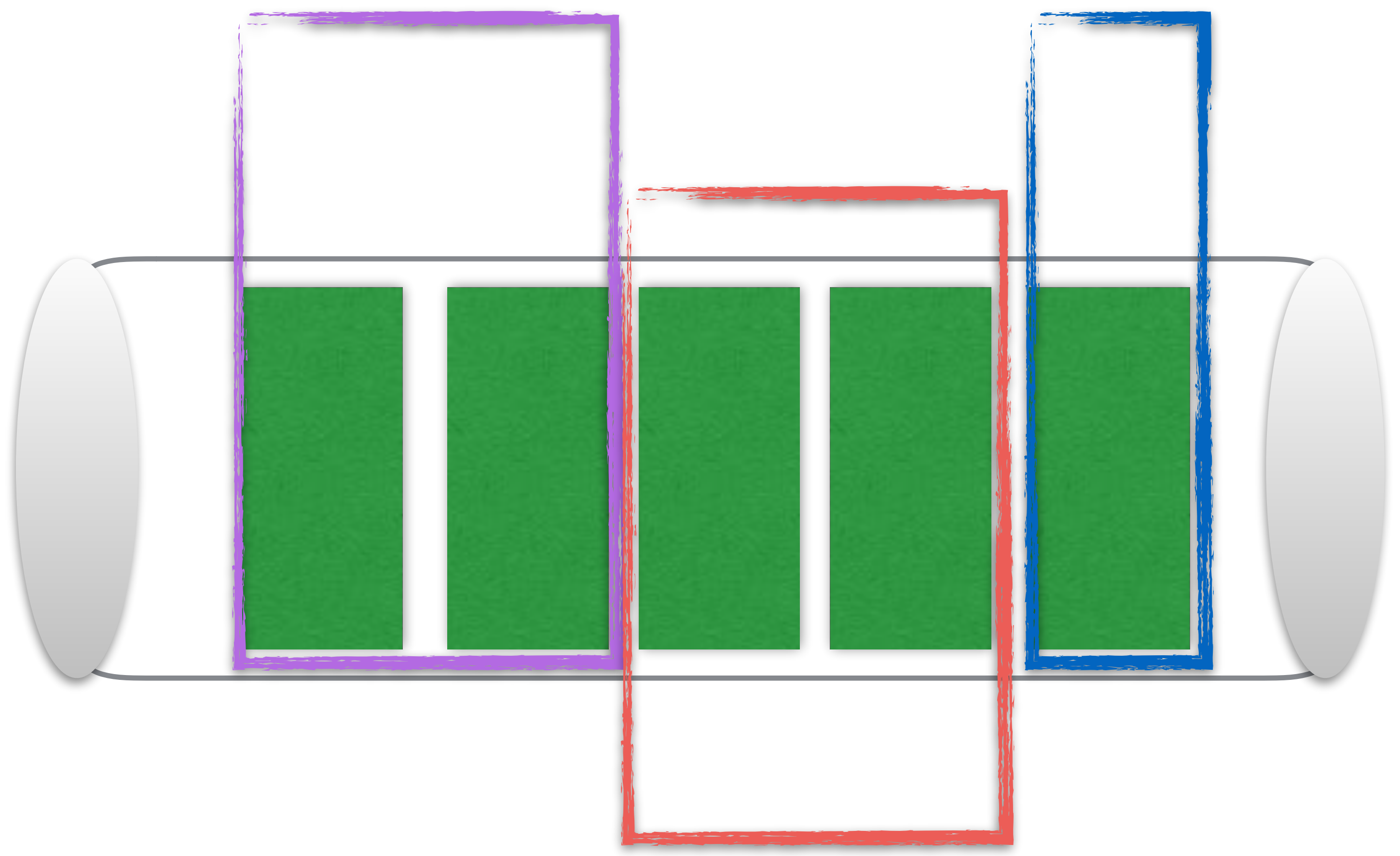
Apache Kafka Cluster



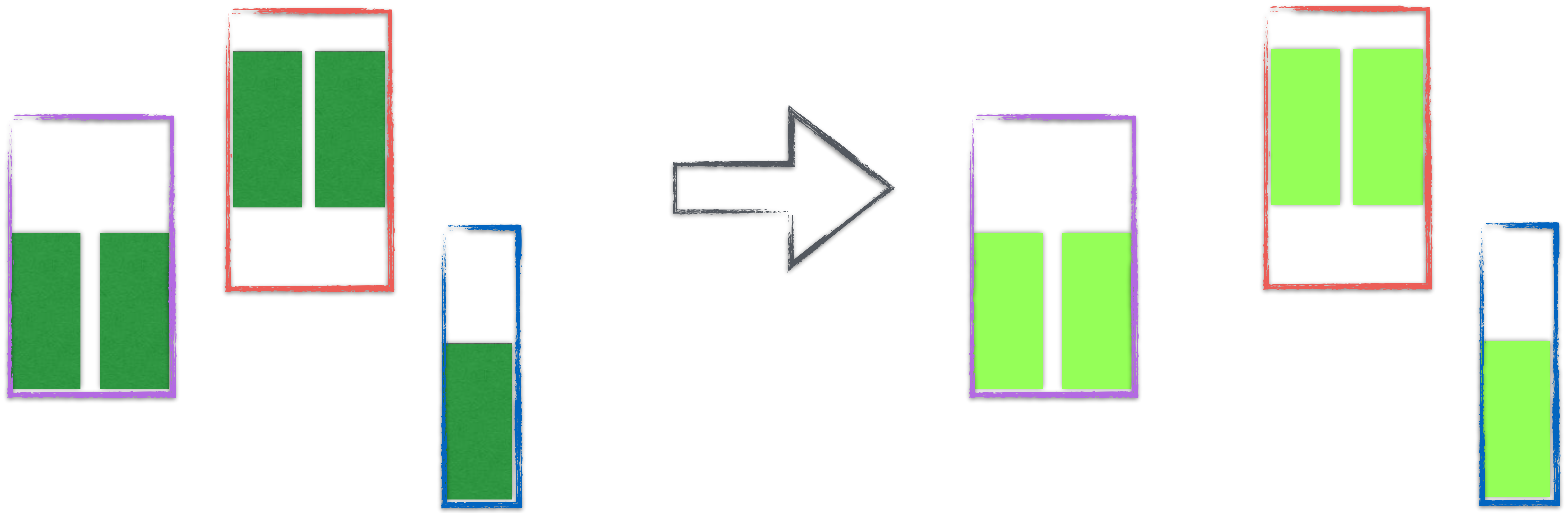
Apache Kafka Topic



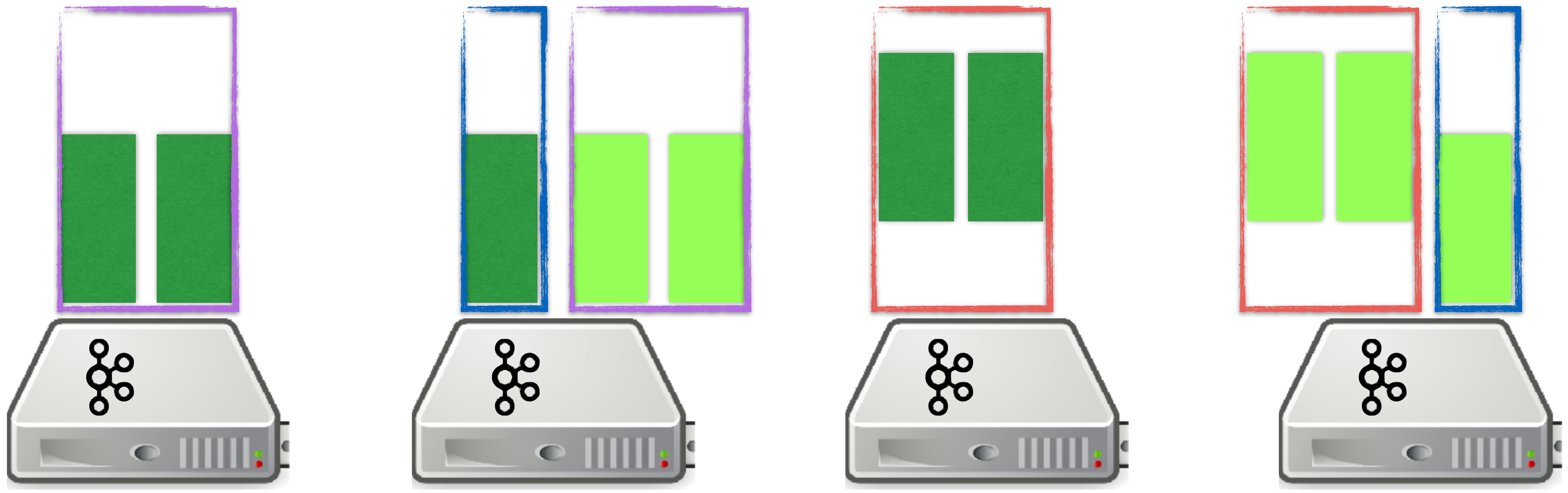
Apache Kafka Partitions



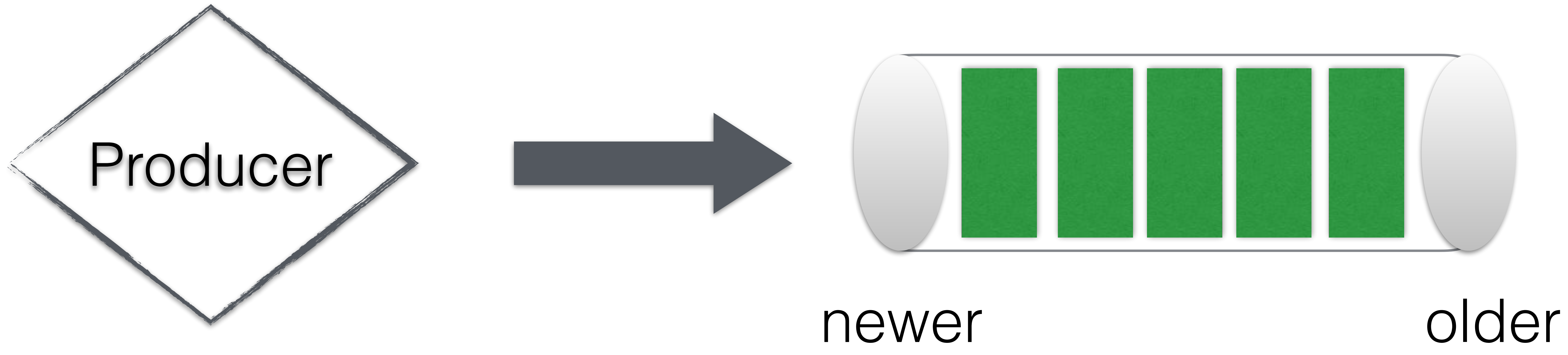
Apache Kafka Replications



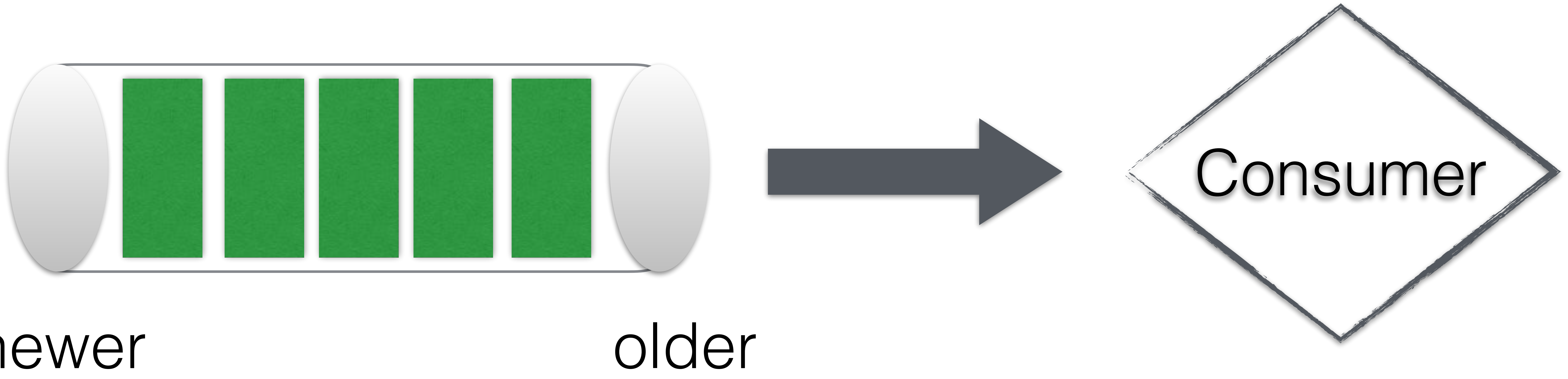
Apache Kafka Distribute Partitions



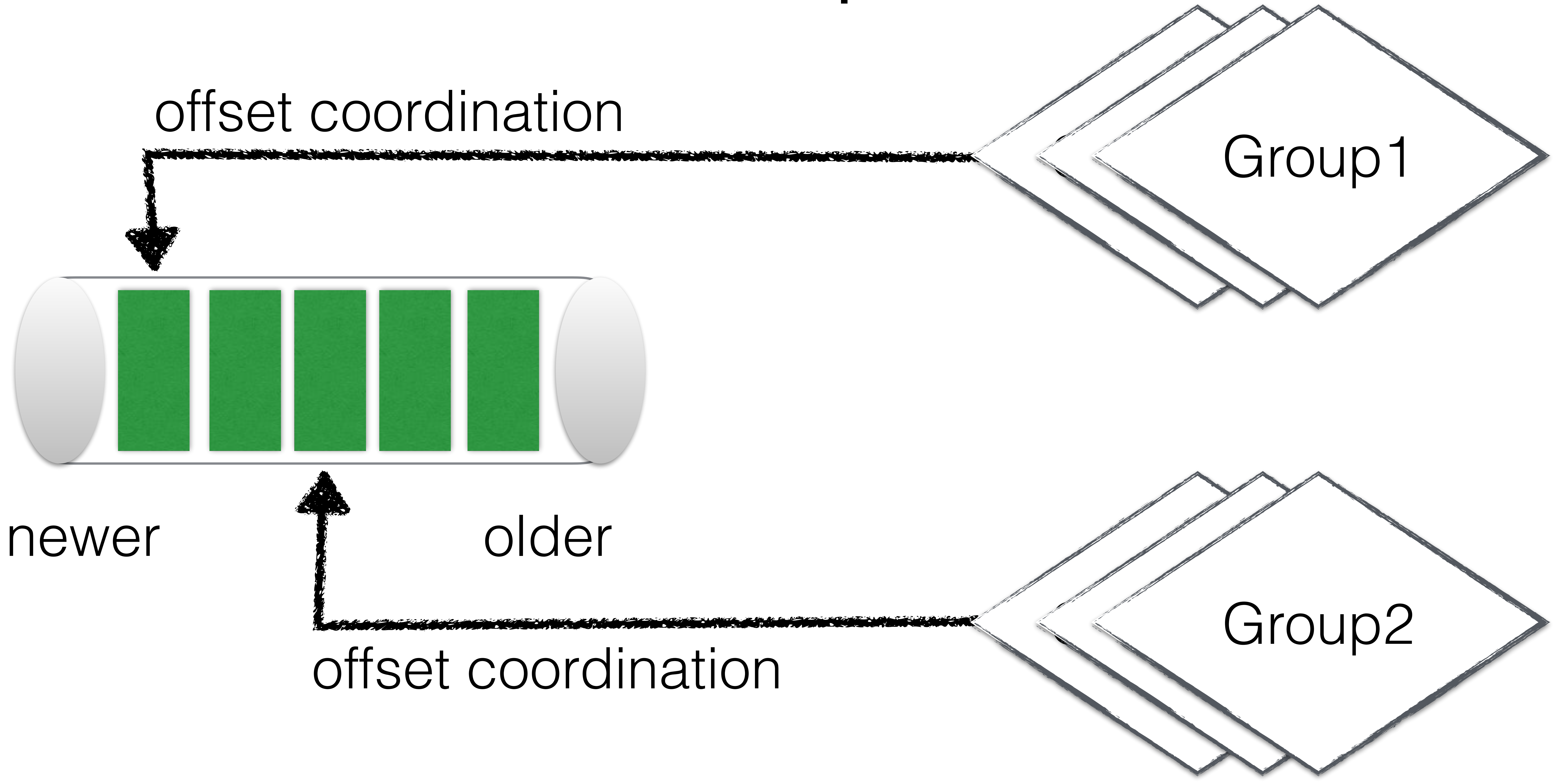
Apache Kafka Producer



Apache Kafka Consumer



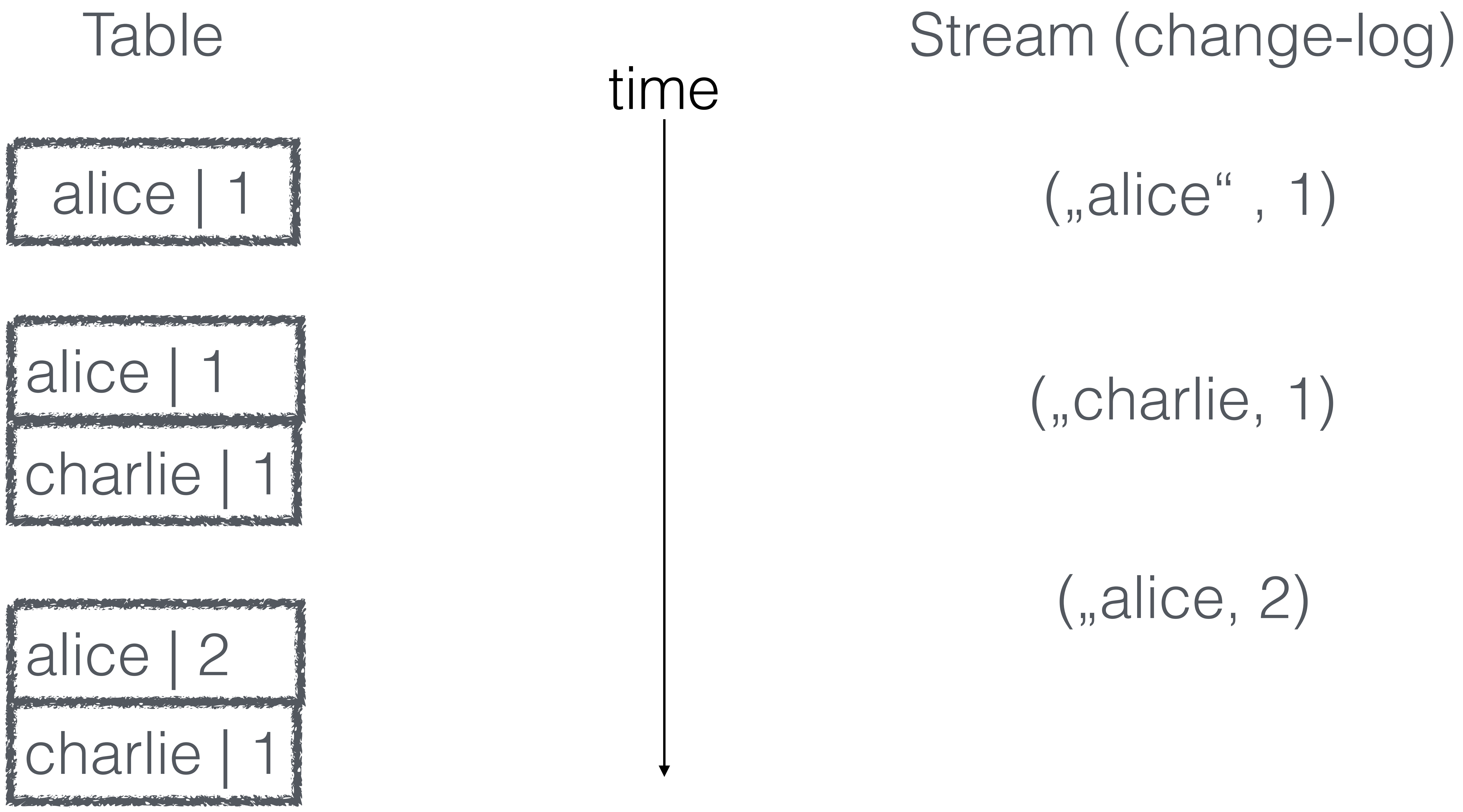
Apache Kafka Consumer Groups



Apache Kafka Streams

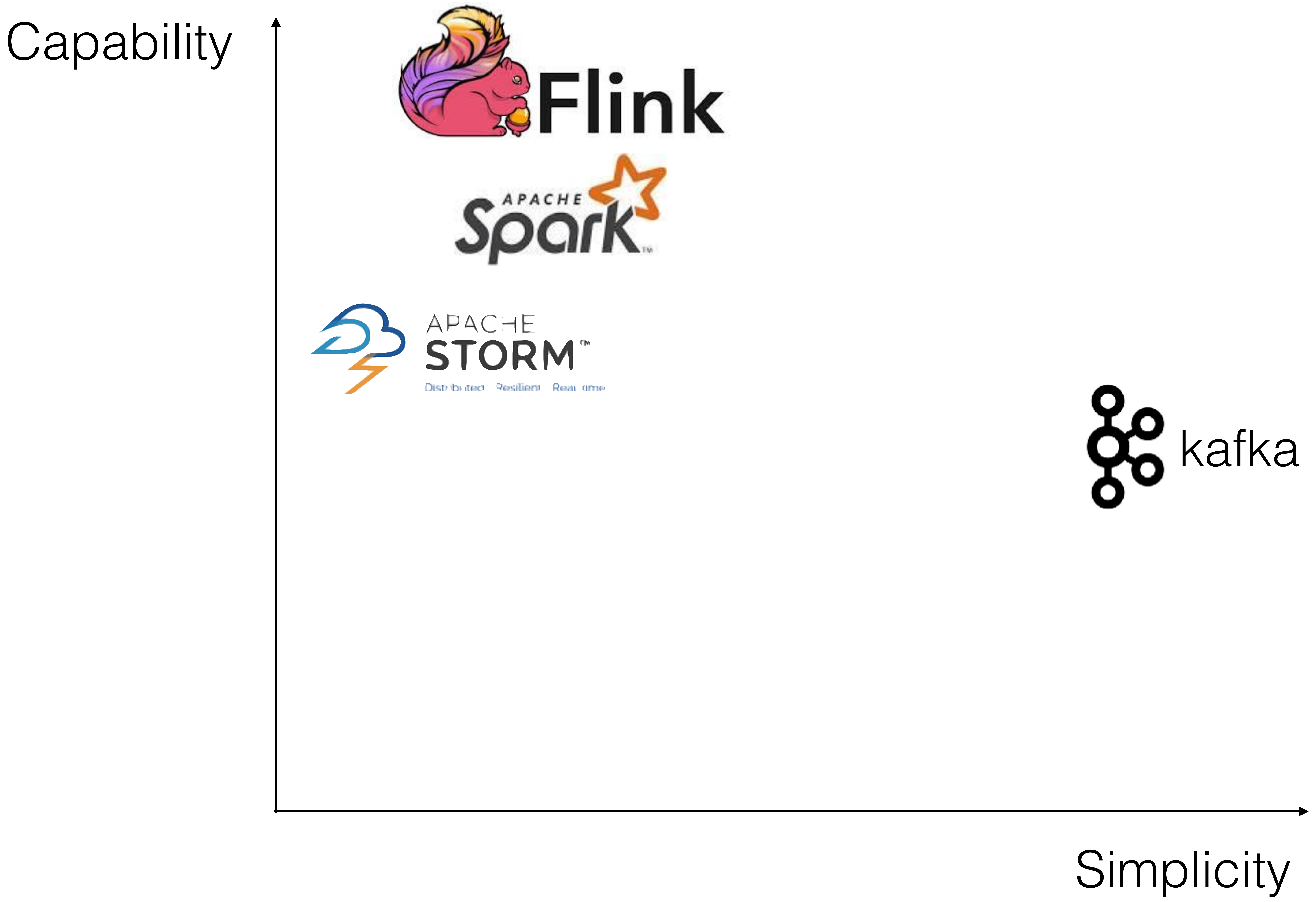
Low-Level-API	High-Level-API
Topology Builder	Stream and Table
Custom Aggregators	Simple Transformation
Custom Processors	Simple Joins

Kafka Streaming API



Stream processing

Background


Capability
Flink
APACHE Spark
APACHE STORM
Distributed Resilient Real time
kafka
Simplicity

Background

kafka streams

APACHE Spark

Flink

kafka

~~hadoop~~

DISTRIBUTED PIPE’ING

Kafka featuring Unix

|

```
(split %1 "\s")
```

Kafka featuring Unix

|

`(split %1 "\s")`

Message Broker

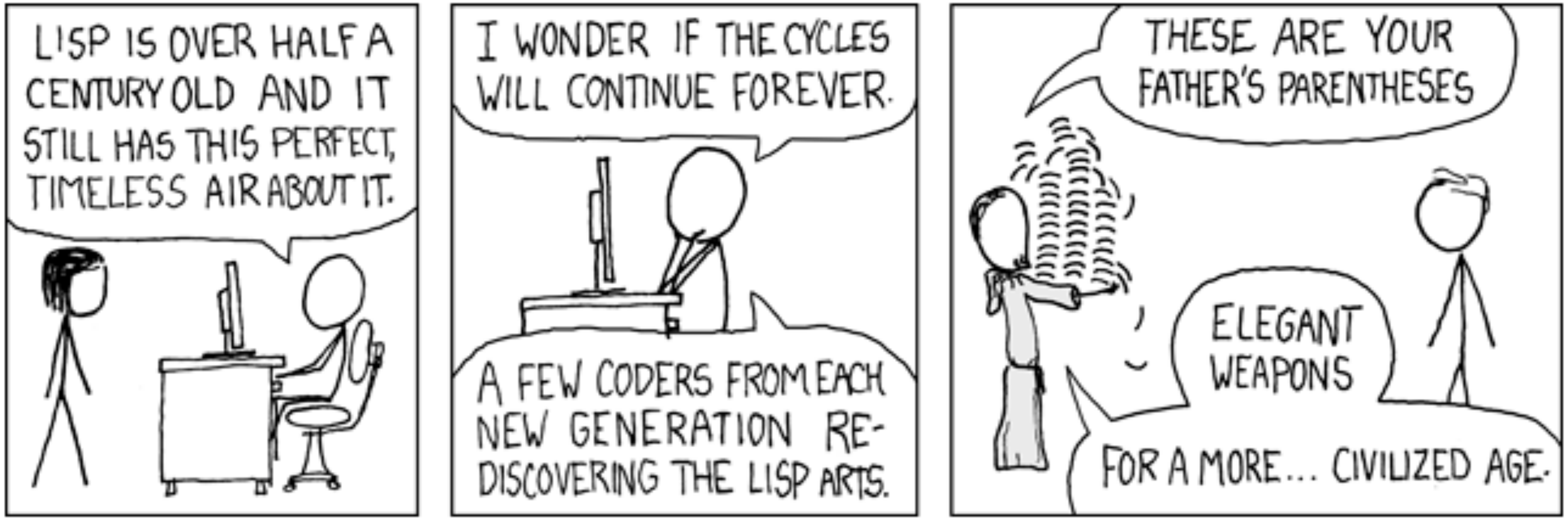
Stream processing job

Kafka featuring Unix

```
(split %1 "\s") | (sketch %1) | (agg-by-key %1) | (store %1)
```

Distributed Tools

Distributed Tools


LISP IS OVER HALF A CENTURY OLD AND IT STILL HAS THIS PERFECT, TIMELESS AIR ABOUT IT.
I WONDER IF THE CYCLES WILL CONTINUE FOREVER.
A FEW CODERS FROM EACH NEW GENERATION RE-DISCOVERING THE LISP ARTS.
THESE ARE YOUR FATHER'S PARENTHESES
ELEGANT WEAPONS
FOR A MORE... CIVILIZED AGE.

https://xkcd.com/297/

CUTTER

Distributed Tools

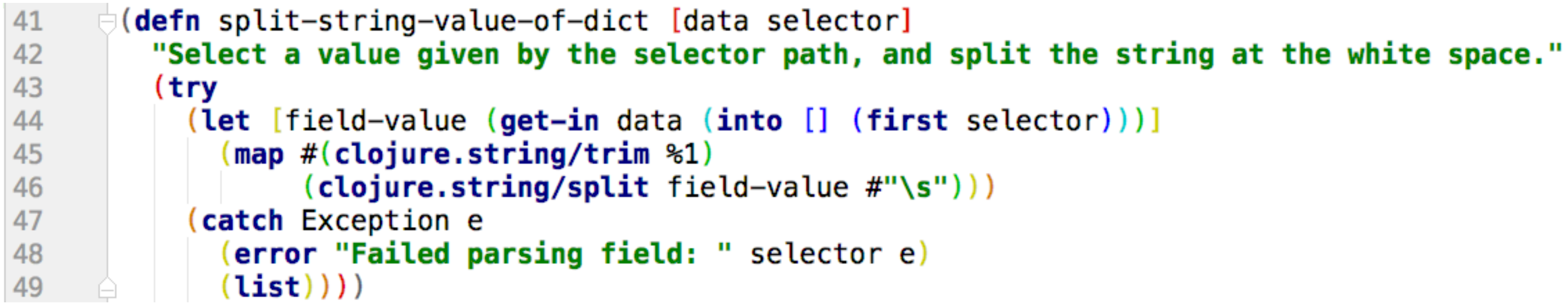
Input:

Edmilson Alves 0 Edmilson Alves -LRB- born February 17 , 1976 -RRB- , is a Brazilian midfielder who currently plays for Roasso Kumamoto in the J. League Division 2 .

Output:

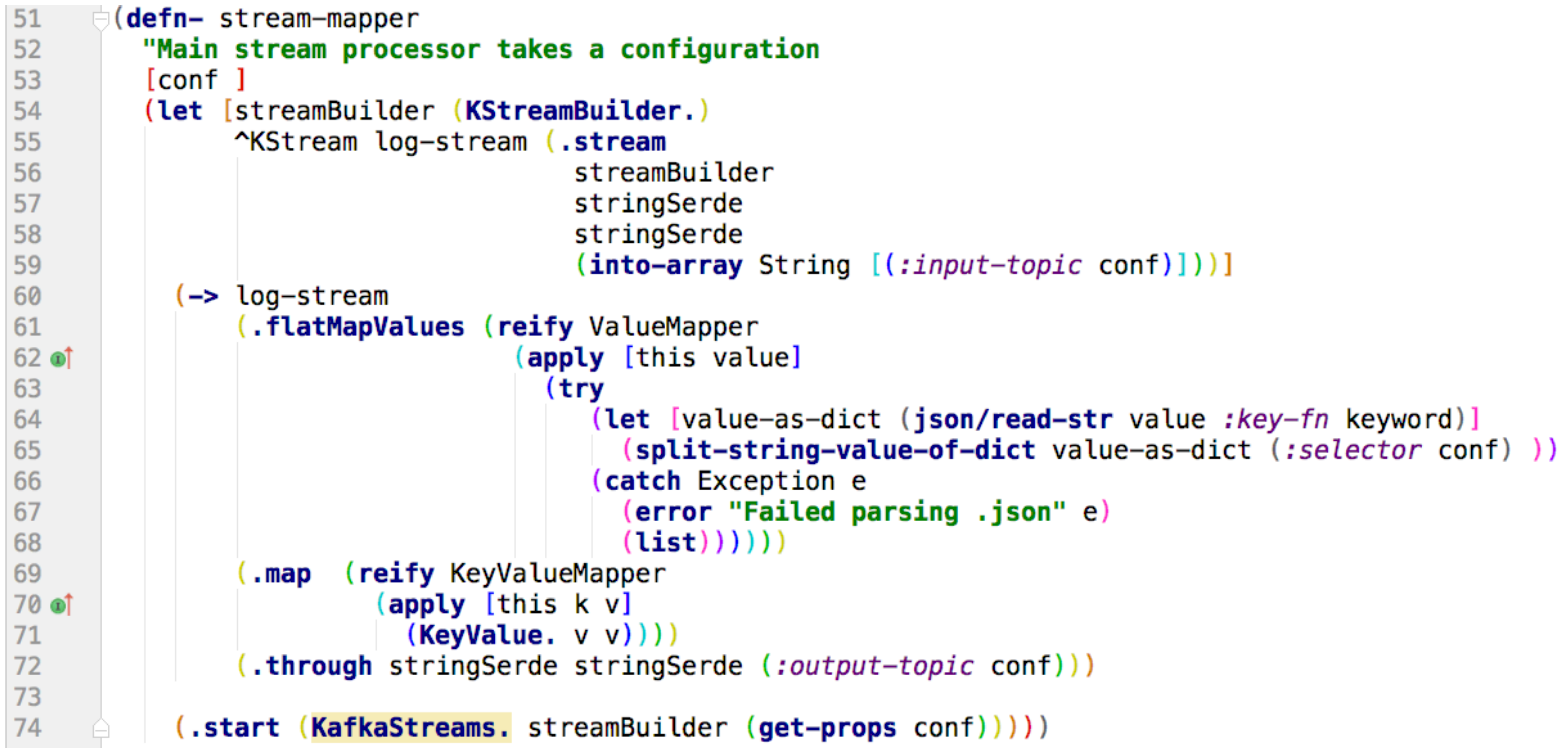
[Edmilson, Alves, 0, Edmilson, Alves, LRB, born **...]**

Distributed Tools



```
(defn split-string-value-of-dict [data selector]
  "Select a value given by the selector path, and split the string at the white space."
  (try
    (let [field-value (get-in data (into [] (first selector)))]
      (map #(clojure.string/trim %1)
           (clojure.string/split field-value #"\s")))
    (catch Exception e
      (error "Failed parsing field: " selector e)
      (list))))
```

Distributed Tools



```clojure
(defn- stream-mapper
  "Main stream processor takes a configuration
  [conf ]
  (let [streamBuilder (KStreamBuilder.)
        ^KStream log-stream (.stream
                              streamBuilder
                              stringSerde
                              stringSerde
                              (into-array String [(:input-topic conf)]))]
    (-> log-stream
        (.flatMapValues (reify ValueMapper
                          (apply [this value]
                            (try
                              (let [value-as-dict (json/read-str value :key-fn keyword)]
                                (split-string-value-of-dict value-as-dict (:selector conf) ))
                              (catch Exception e
                                (error "Failed parsing .json" e)
                                (list))))))
        (.map  (reify KeyValueMapper
                 (apply [this k v]
                   (KeyValue. v v))))
        (.through stringSerde stringSerde (:output-topic conf)))

    (.start (KafkaStreams. streamBuilder (get-props conf)))))
```

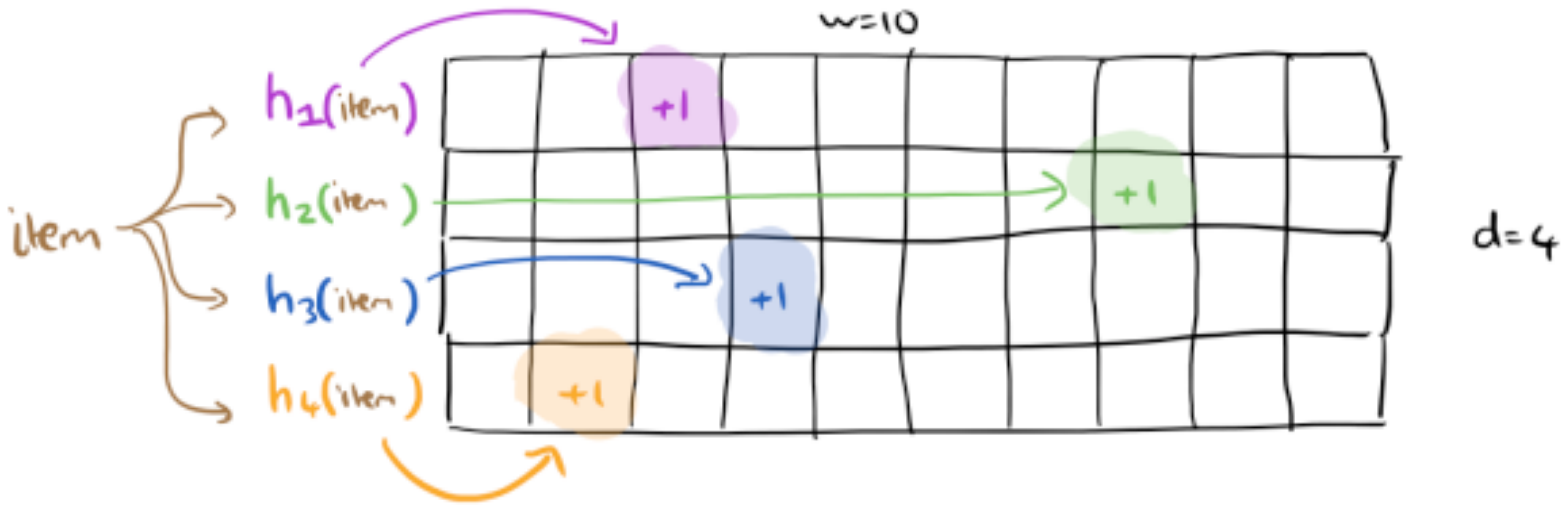
Heavy Hitter

clj-kstream-hh

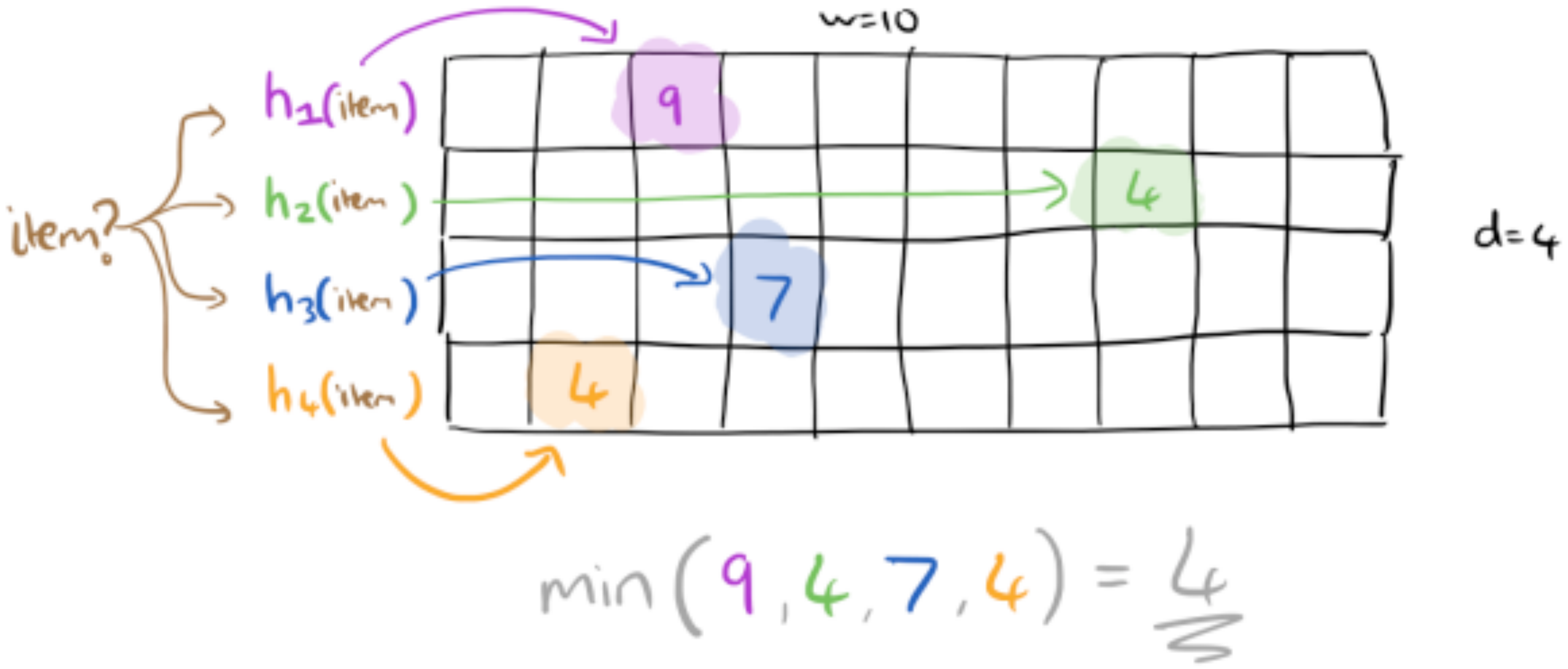
Input: [Edmilson, Alves, 0, Edmilson, Alves, LRB, born, …]

Output: Edmilson ~10 Alves ~8

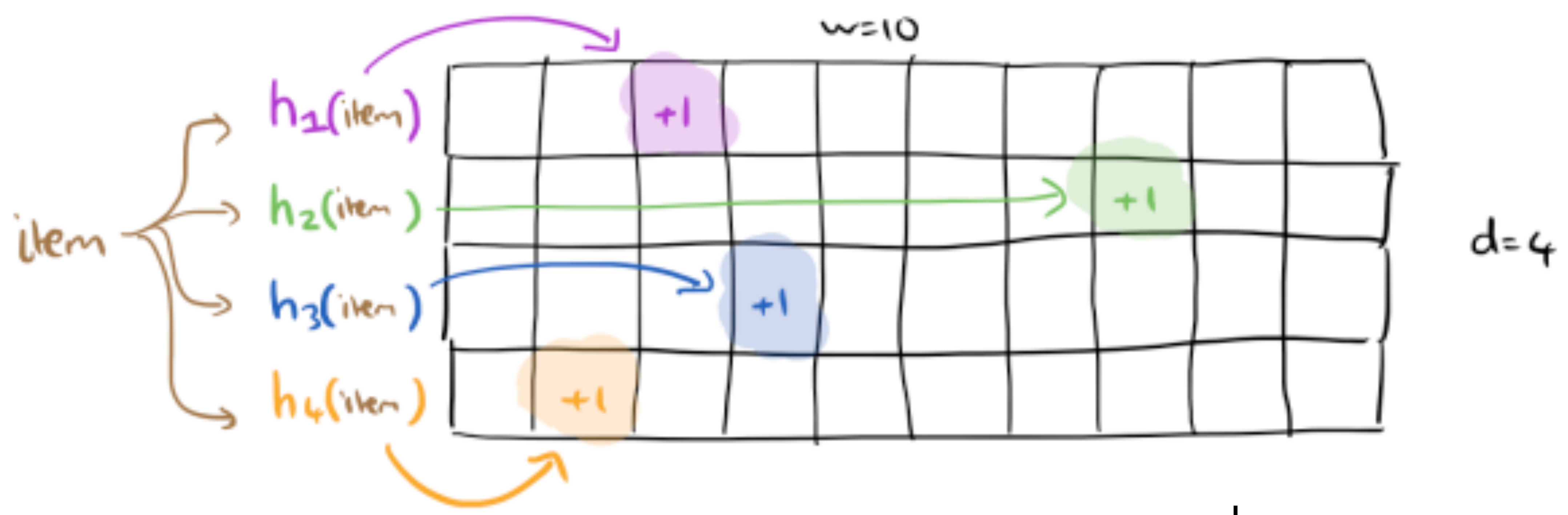
Count Min (CM) sketch



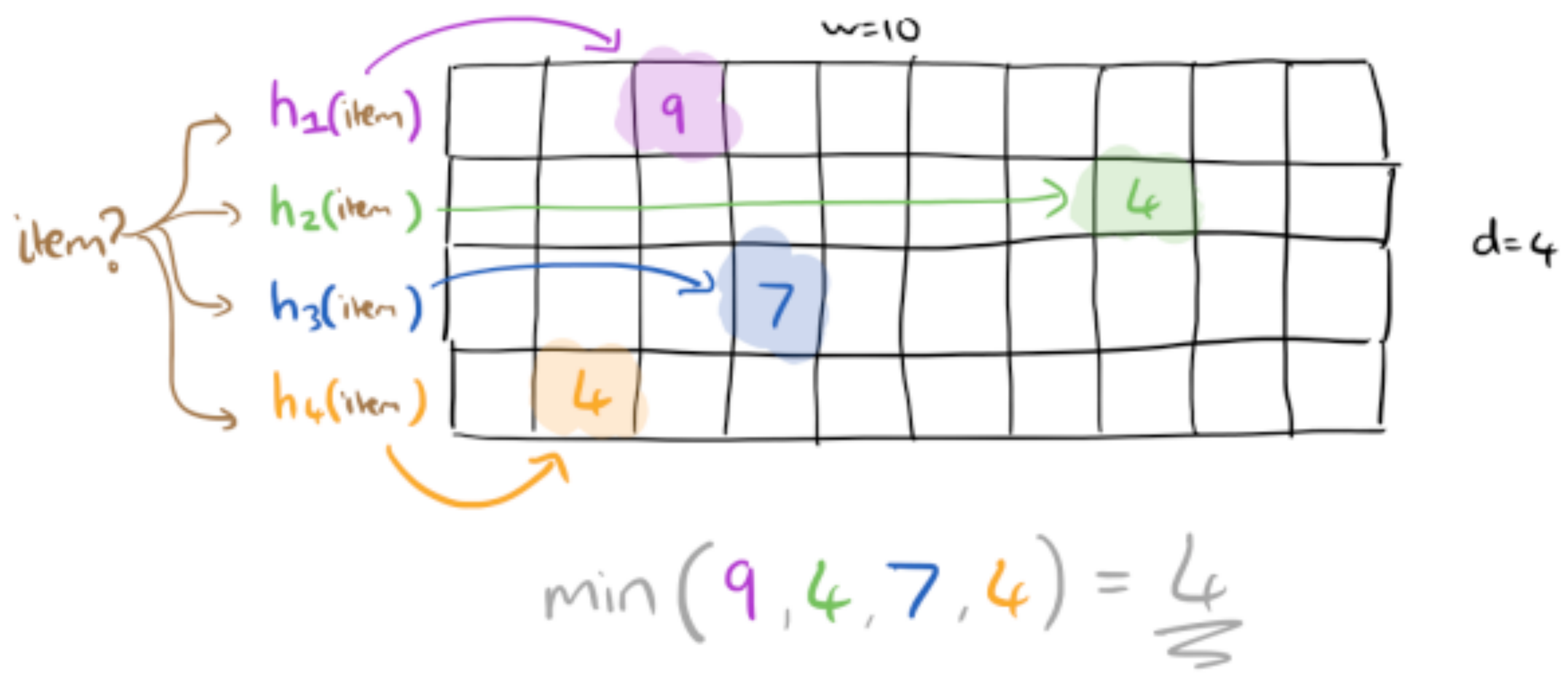
CM sketch retrieval



Distributed Tools



retrieve sketched value



take top_n

Heavy Hitter for t_1

And ~10
Bob ~7
Alice ~5
Foo ~3
Bar ~2

Topology

```
(defn- heavy-hitter-processor
 "Main stream processor takes a configuration and a mapper function to apply."
 [conf]
 (let [streamBuilder (-> (new TopologyBuilder)
                    (.addSource (:name conf) string_dser string_dser  (into-array [(:input-topic conf)]))
                    (.addProcessor "HeavyHitter"
                                (reify ProcessorSupplier
                                  (get [this]
                                    (get-processor)))
                                (into-array [(:name conf)]))
                    (.addStateStore
                      (->> (Stores/create storeName)
                          (.withStringKeys)
                          (.withLongValues)
                          (.inMemory)
                          (.build))
                      (into-array ["HeavyHitter"]))
                    (.addSink
                      "Sink"
                      (:output-topic conf)
                      string_ser
                      string_ser
                      (into-array ["HeavyHitter"])))]
   (.start
     (KafkaStreams.
       streamBuilder
       (get-props conf)))))
```

Processor

```
(defn ^Processor get-processor []
     (reify org.apache.kafka.streams.processor.Processor
      (init [this context]
       (.schedule (:context @application-state) (:time-window @application-state))
       (swap! hh/state assoc
             :top-n 5
             :number-of-hashfn 10N
             :bucket-size 1000N)
       (reset! hh/hitter ^(priority-map))
       (reset! hh/min-sketch (make-array Integer/TYPE 10N 1000N)) …)

      (process [this key value]
       (debug "Process (k,v)::" key value)
       (hh/sketch-value value)
       (hh/add-to-hitter value) …)

      (punctuate [this timestamp …)

      (close [this]
       (.close (:store @application-state)))))
```

Window Aggregate

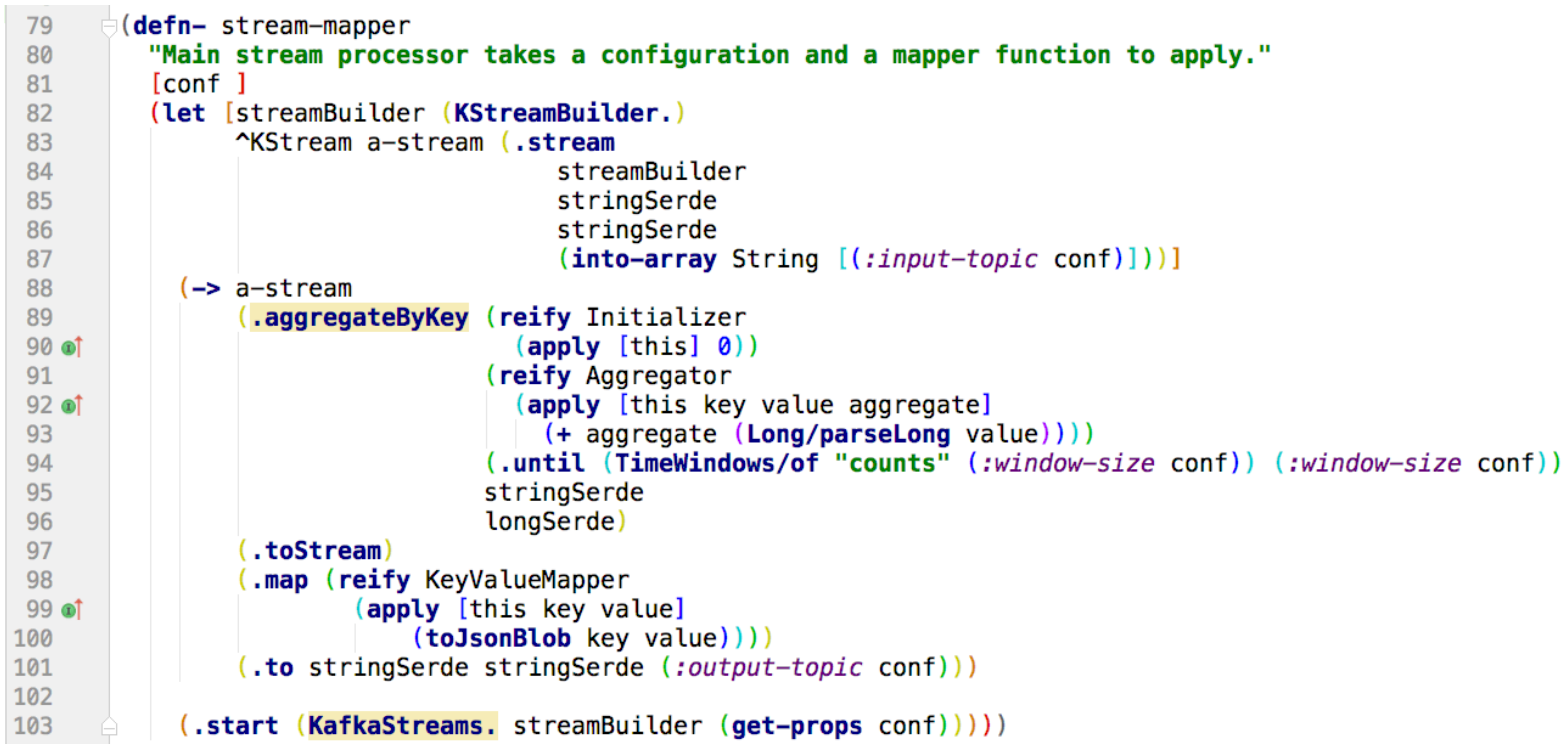
Distributed Tools

(key, value)

Input: Alves ~8 Alves ~10 Edmilson ~5 Edmilson~3

Output: Alves ~18 Edmilson ~8

Distributed Tools



```clojure
(defn- stream-mapper
  "Main stream processor takes a configuration and a mapper function to apply."
  [conf ]
  (let [streamBuilder (KStreamBuilder.)
        ^KStream a-stream (.stream
                           streamBuilder
                           stringSerde
                           stringSerde
                           (into-array String [(:input-topic conf)]))]
    (-> a-stream
        (.aggregateByKey (reify Initializer
                           (apply [this] 0))
                         (reify Aggregator
                           (apply [this key value aggregate]
                             (+ aggregate (Long/parseLong value))))
                         (.until (TimeWindows/of "counts" (:window-size conf)) (:window-size conf))
                         stringSerde
                         longSerde)
        (.toStream)
        (.map (reify KeyValueMapper
                (apply [this key value]
                  (toJsonBlob key value))))
        (.to stringSerde stringSerde (:output-topic conf)))

    (.start (KafkaStreams. streamBuilder (get-props conf)))))
```

ELASTICSEARCH SINK

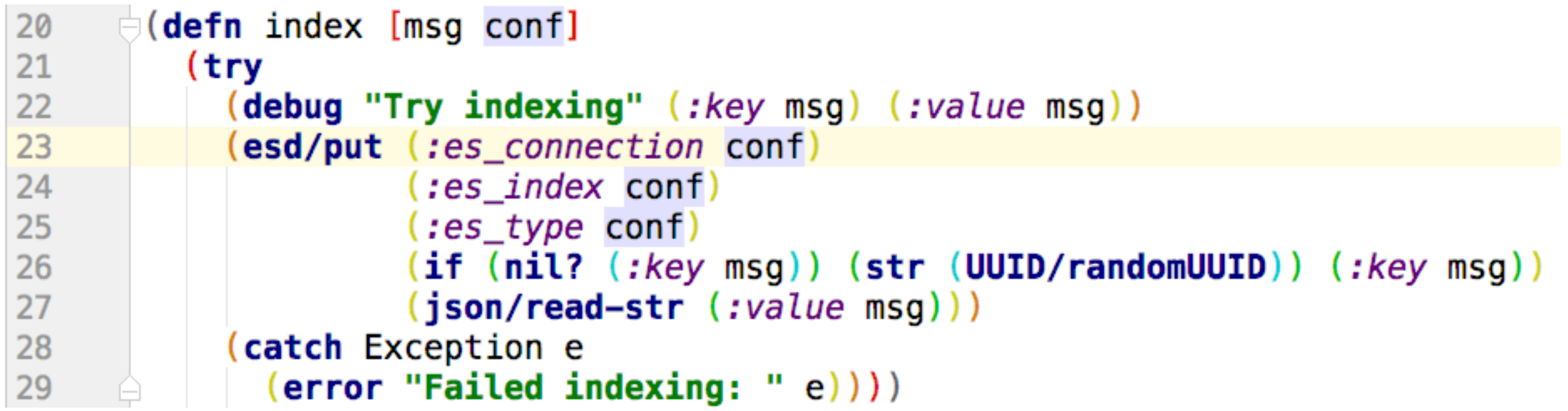
Distributed Tools

Input: Alves ~18

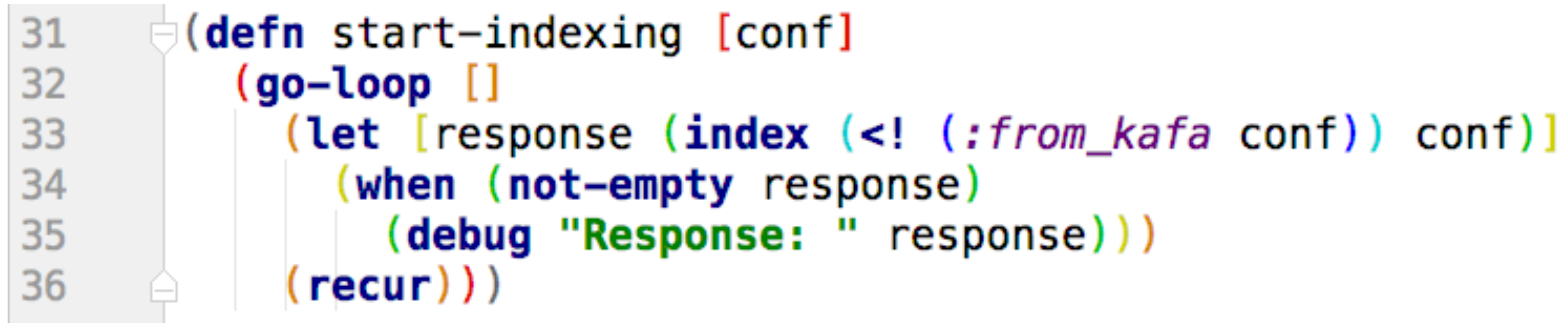
Output:

```
{
“name“: “Alves“,
“count“:  18,
“time“: “January 26th 2017, 17:03:00.000”
}
```

Distributed Tools



```
(defn index [msg conf]
  (try
    (debug "Try indexing" (:key msg) (:value msg))
    (esd/put (:es_connection conf)
             (:es_index conf)
             (:es_type conf)
             (if (nil? (:key msg)) (str (UUID/randomUUID)) (:key msg))
             (json/read-str (:value msg)))
    (catch Exception e
      (error "Failed indexing: " e))))
```



```
(defn start-indexing [conf]
  (go-loop []
    (let [response (index (<! (:from_kafa conf)) conf)]
      (when (not-empty response)
        (debug "Response: " response)))
    (recur)))
```

almost Kiss

Distributed Tools

Edmilson Alves 0 Edmilson Alves -LRB- born February 17 , 1976 -RRB- , is a Brazilian midfielder who currently plays for Roasso Kumamoto in the J. League Division 2 .

[Edmilson, Alves, 0, Edmilson, Alves, LRB, born **...]**

Alves ~10 Alves ~8

Alves 18

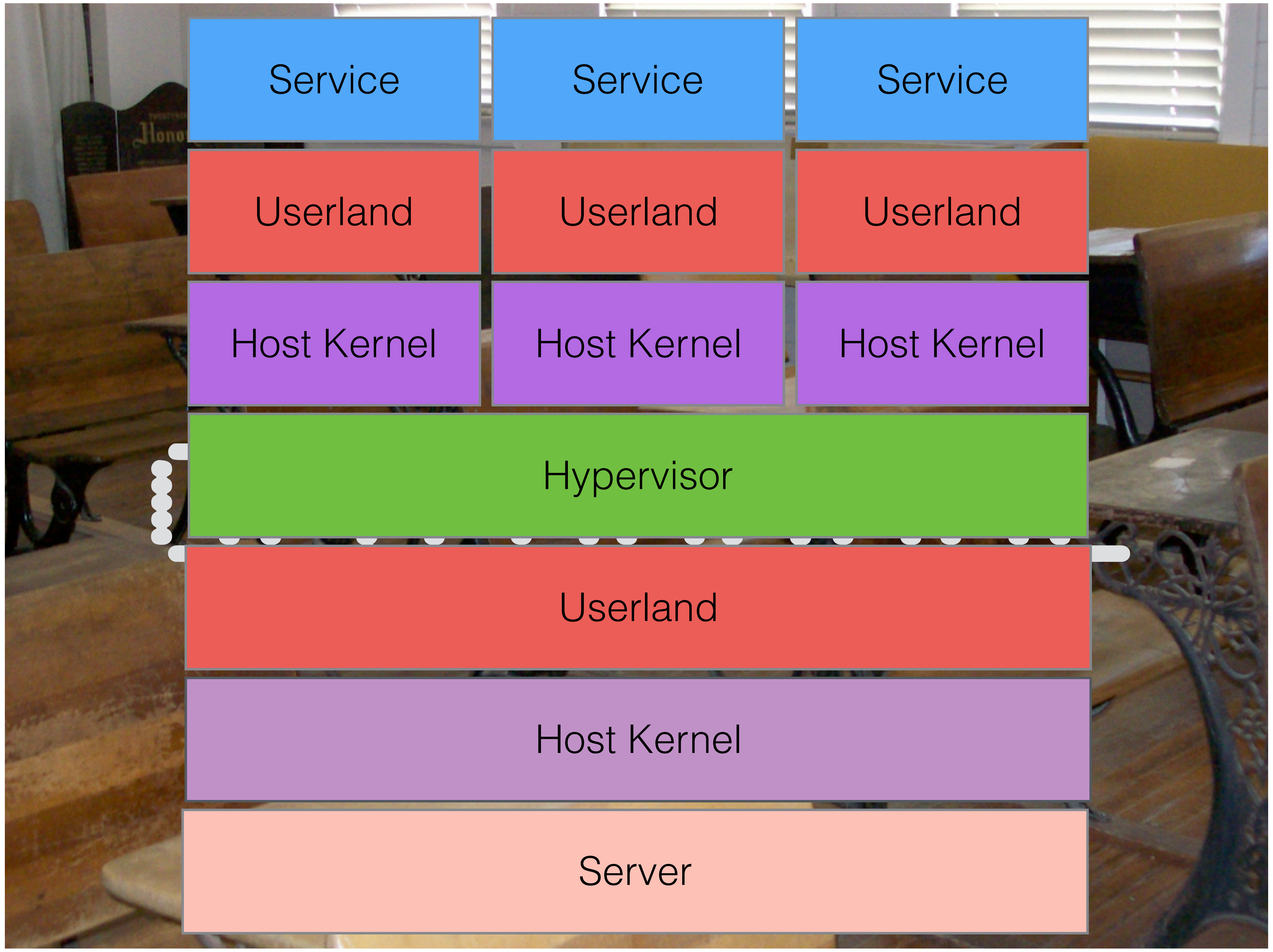
{“name“: “Alves“, “count“: 18, “time“: “January 26th 2017, 17:03:00.000”}

Container

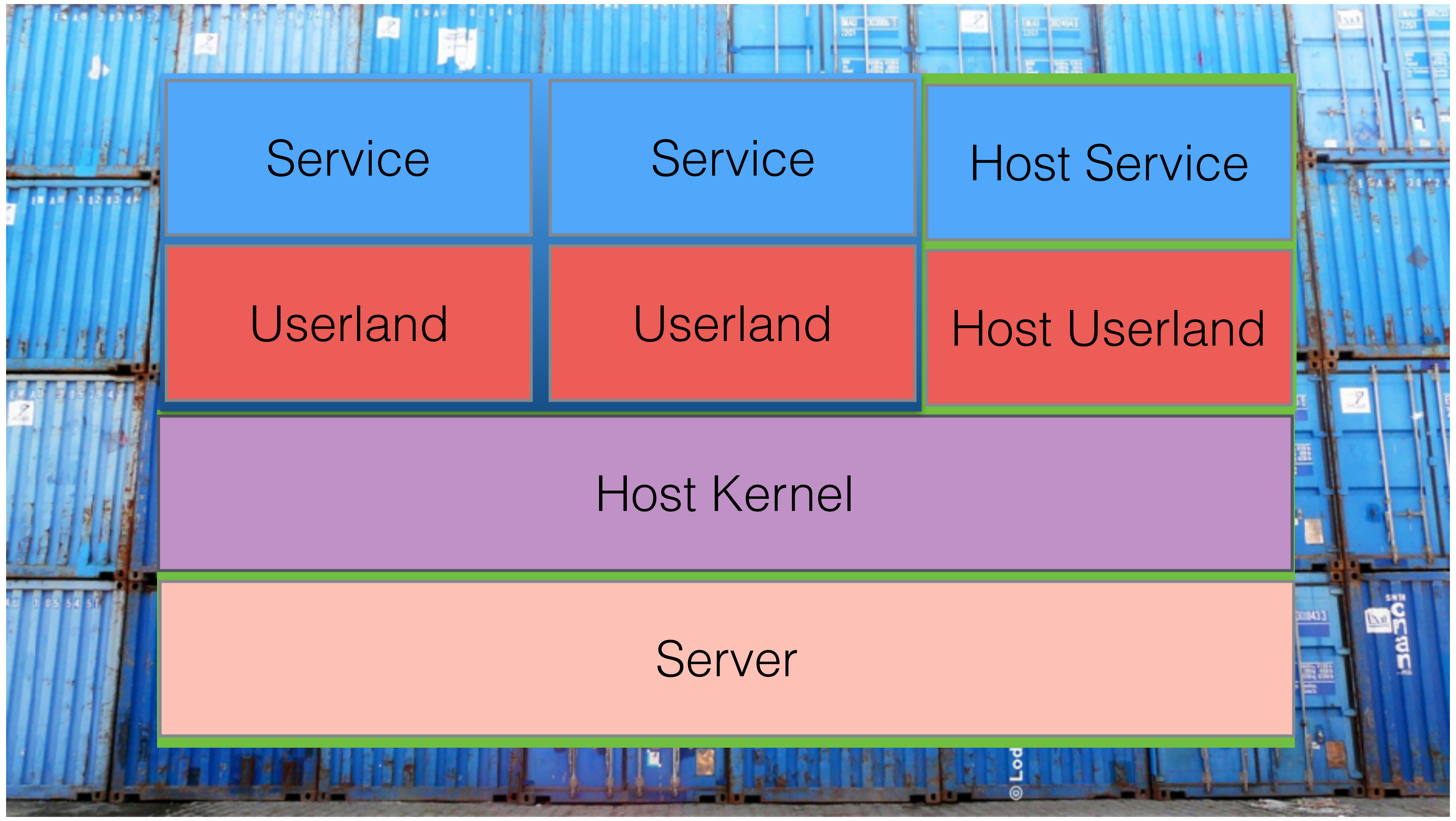
Containers

- […] **Operating-system-level virtualization** is a server virtualization method in which the kernel of an operating system allows the existence of **multiple isolated user-space** instances

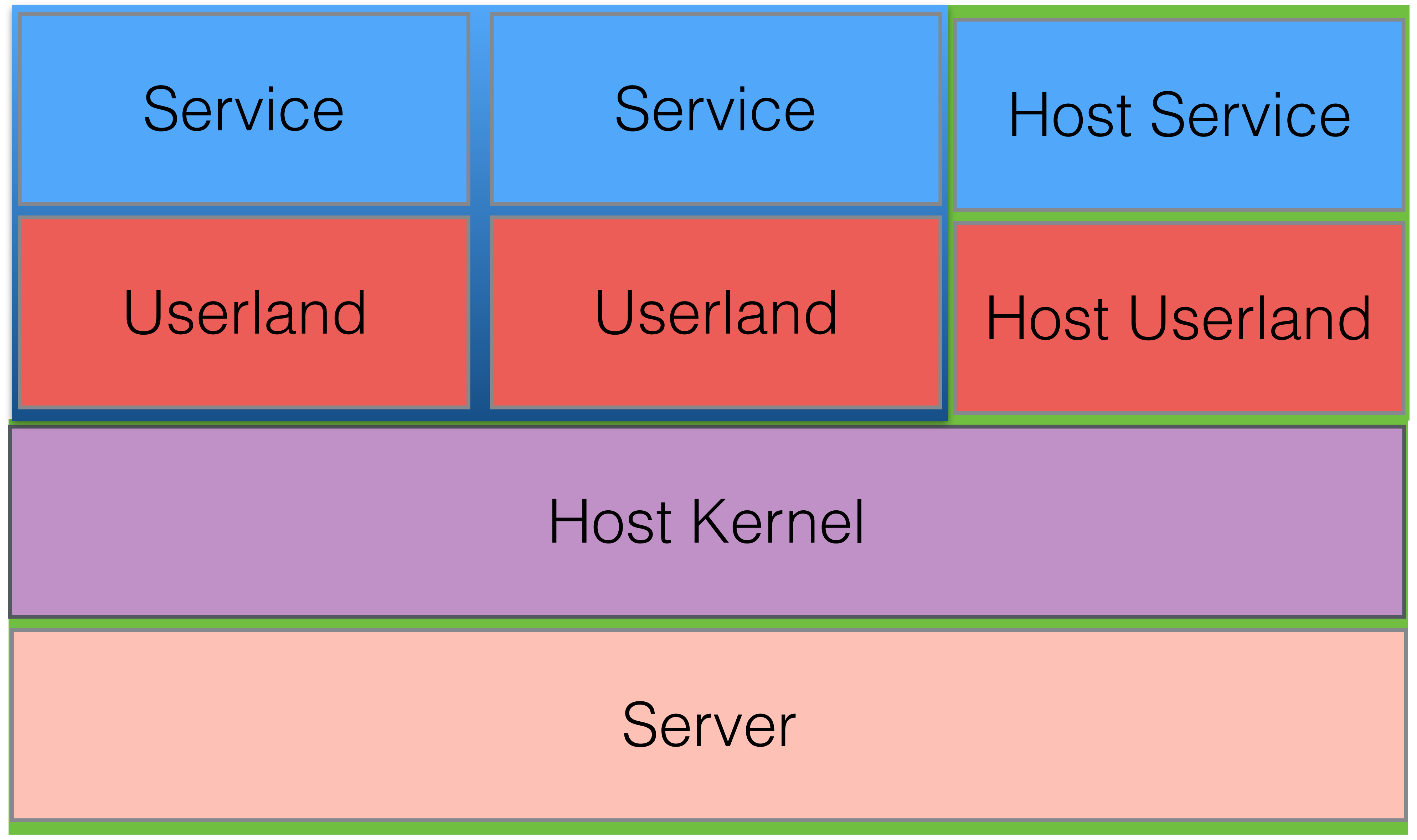
Containers

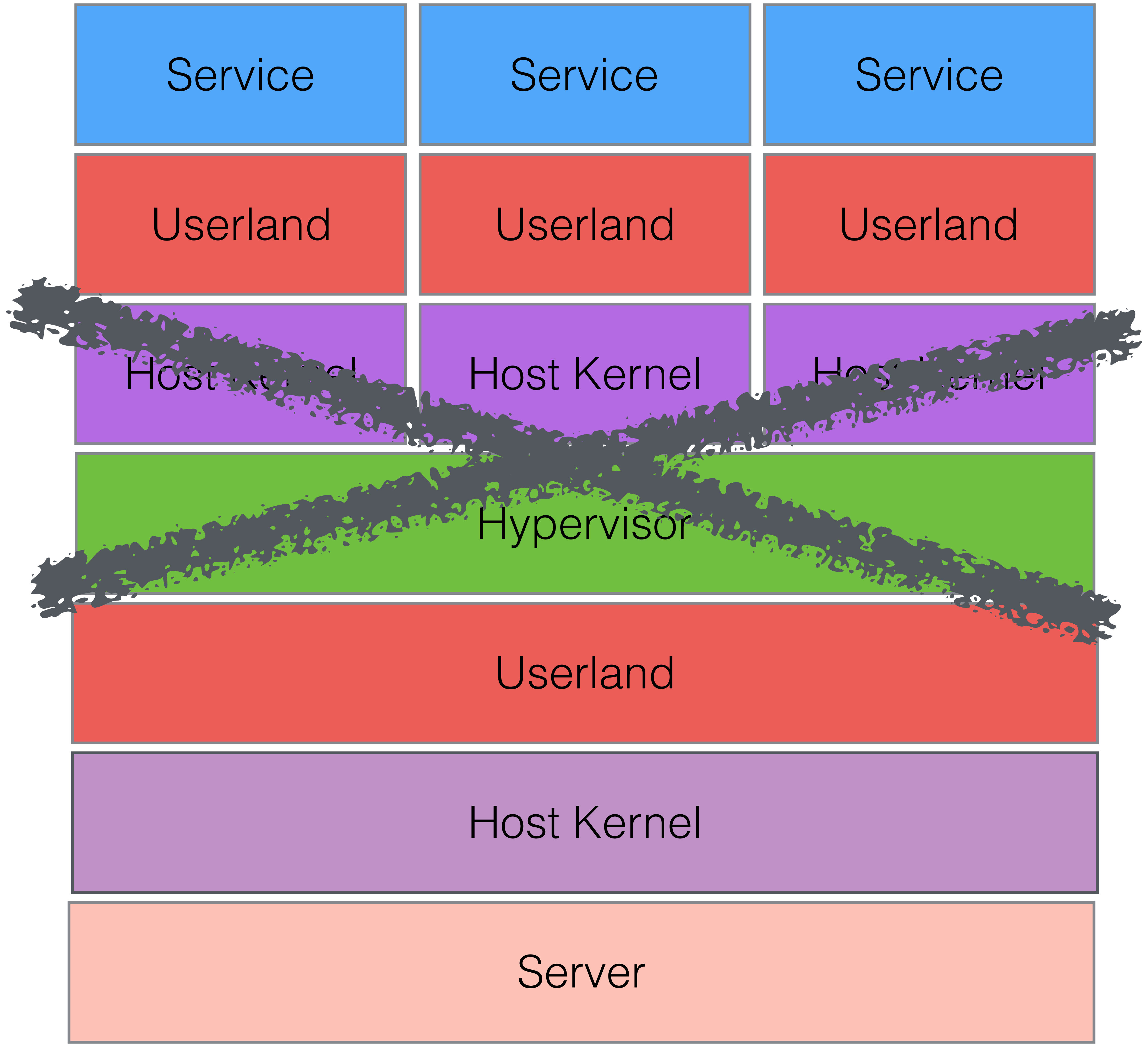

Service
Service
Service
Userland
Userland
Userland
Host Kernel
Host Kernel
Host Kernel
Hypervisor
Userland
Host Kernel
Server

Containers


Service
Service
Host Service
Userland
Userland
Host Userland
Host Kernel
Server

Containers

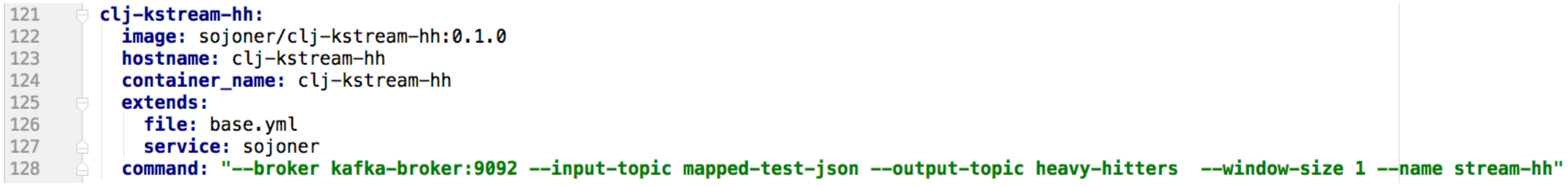

Service
Service
Host Service
Userland
Userland
Host Userland
Host Kernel
Server


Service
Service
Service
Userland
Userland
Userland
Host Kernel
Hypervisor
Userland
Host Kernel
Server

Containers

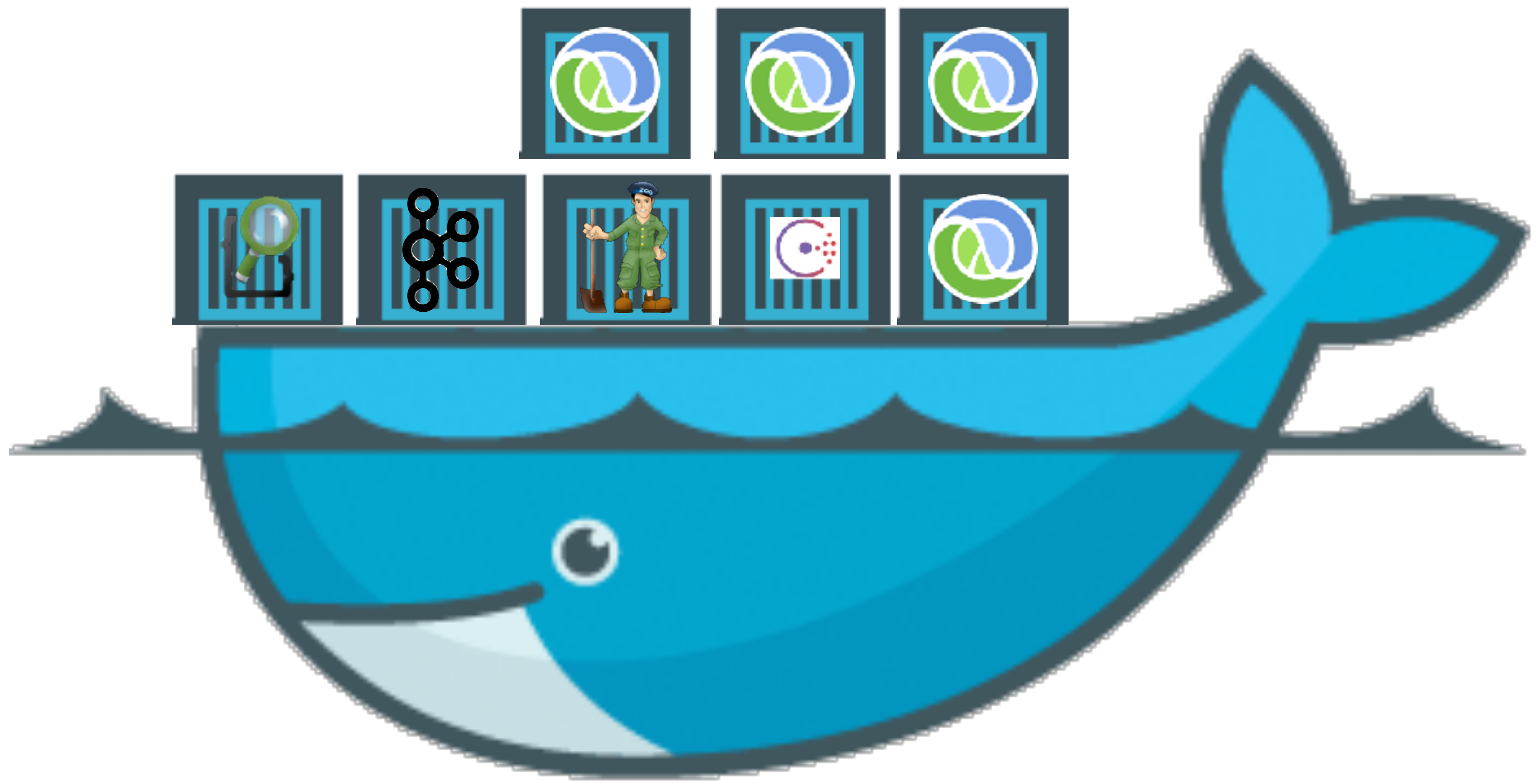
java -jar …args

docker run -t -i CONTAINER-NAME …args



```
clj-kstream-hh:
  image: sojoner/clj-kstream-hh:0.1.0
  hostname: clj-kstream-hh
  container_name: clj-kstream-hh
  extends:
    file: base.yml
    service: sojoner
  command: "--broker kafka-broker:9092 --input-topic mapped-test-json --output-topic heavy-hitters  --window-size 1 --name stream-hh"
```

The development setup…



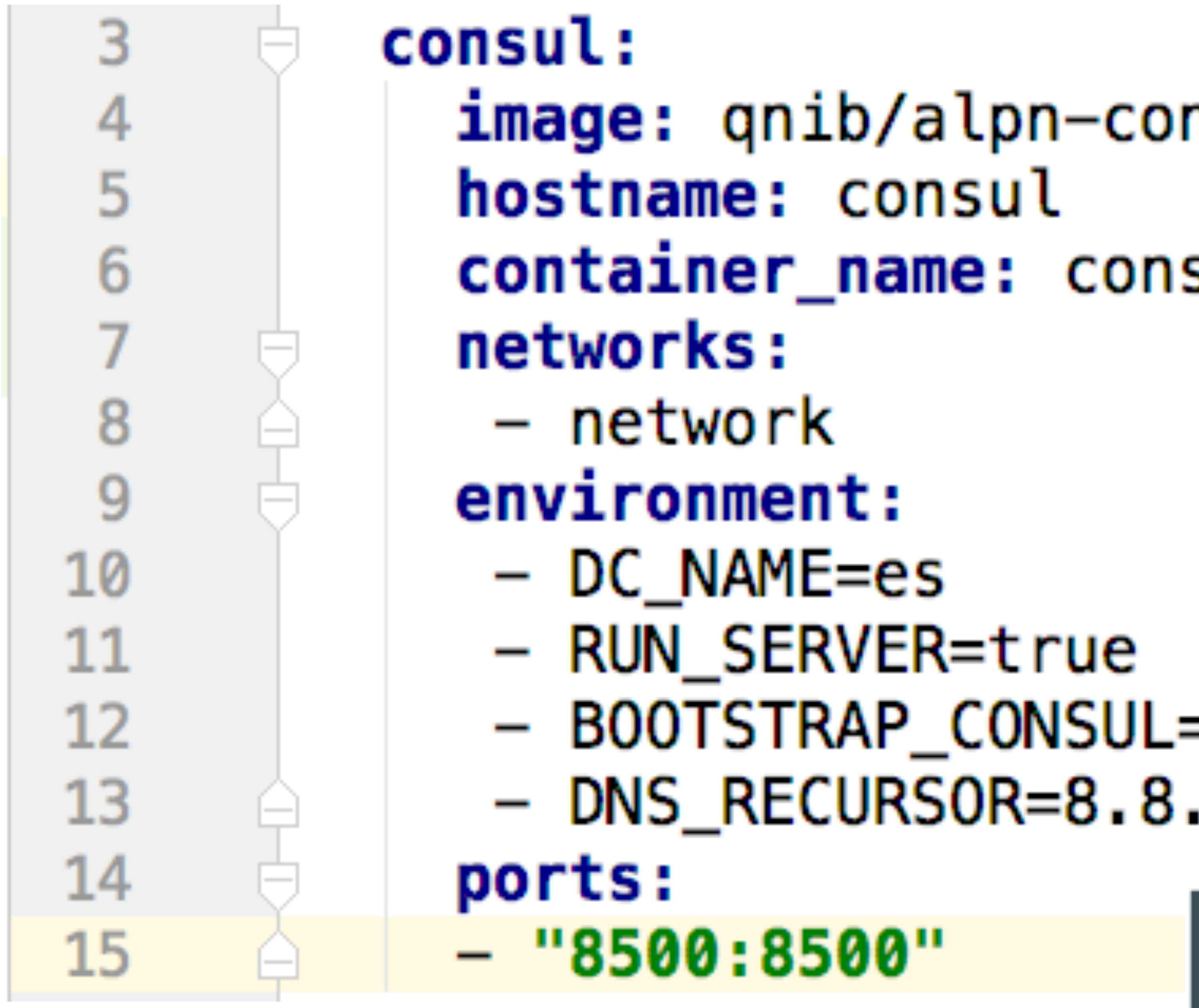
```
$ export DOCKER_HOST=tcp://my.desktop.de:2576
```

THE PIPE IN CONTAINERS

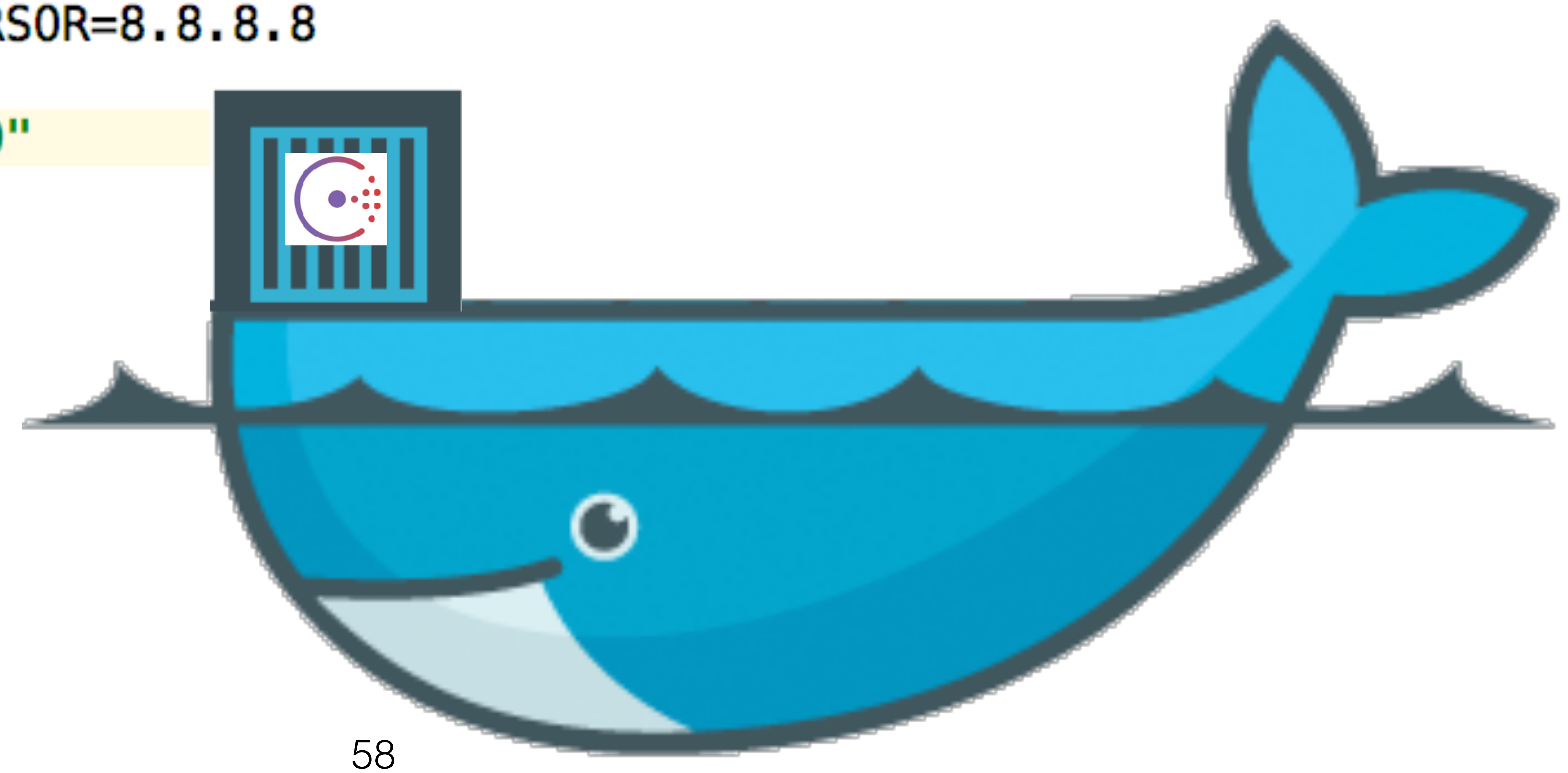
Containers

CONSUL	***Discovering and configuring services in your infrastructure.***
ZooKeeper	***enables highly reliable distributed coordination.***
kafka	***Distributed append log a.k.a Message Broker***
elastic	***Provides a distributed full-text search engine***

Containers

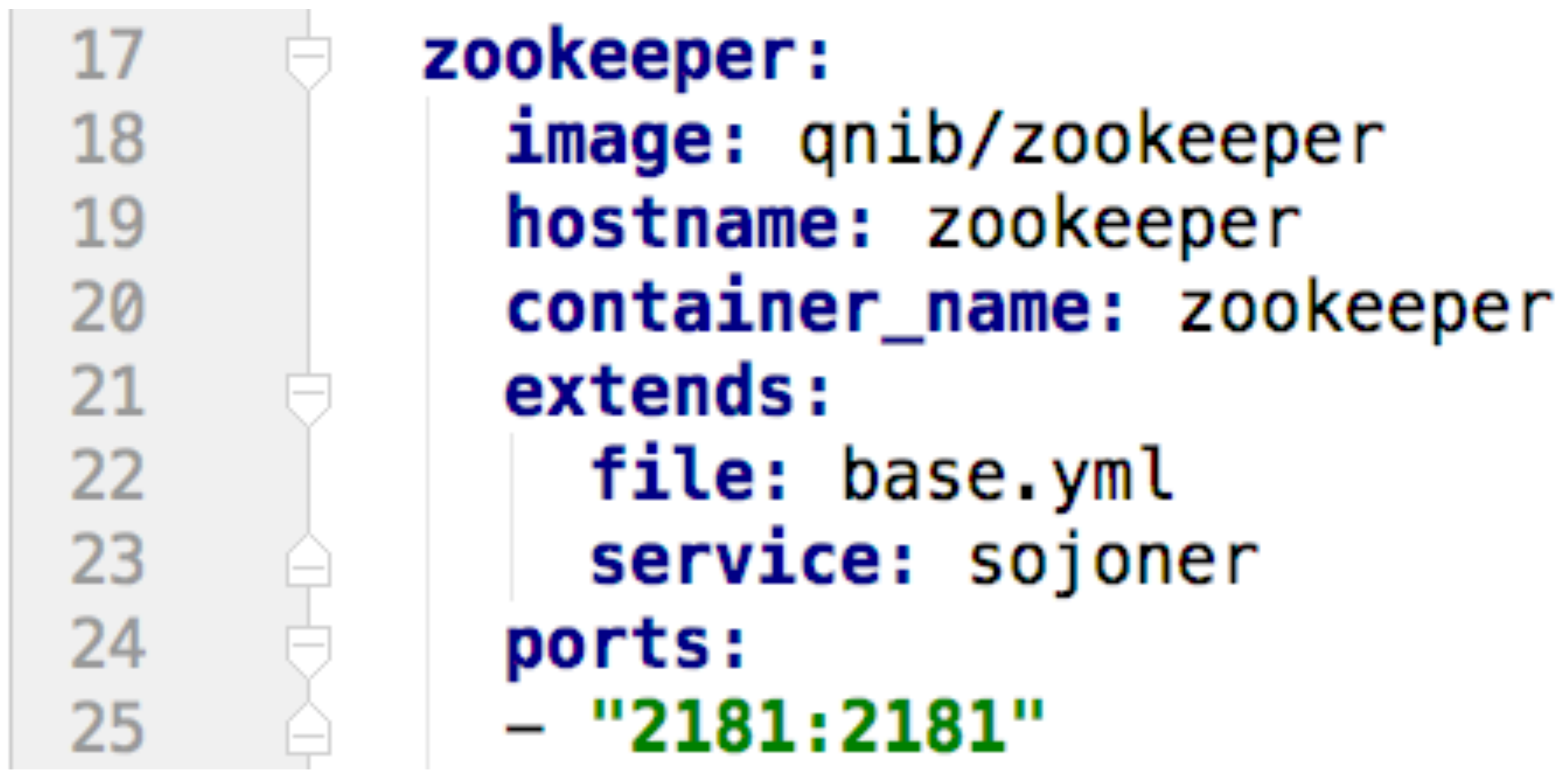


```yaml
consul:
  image: qnib/alpn-consul
  hostname: consul
  container_name: consul
  networks:
   - network
  environment:
   - DC_NAME=es
   - RUN_SERVER=true
   - BOOTSTRAP_CONSUL=true
   - DNS_RECURSOR=8.8.8.8
  ports:
  - "8500:8500"
```

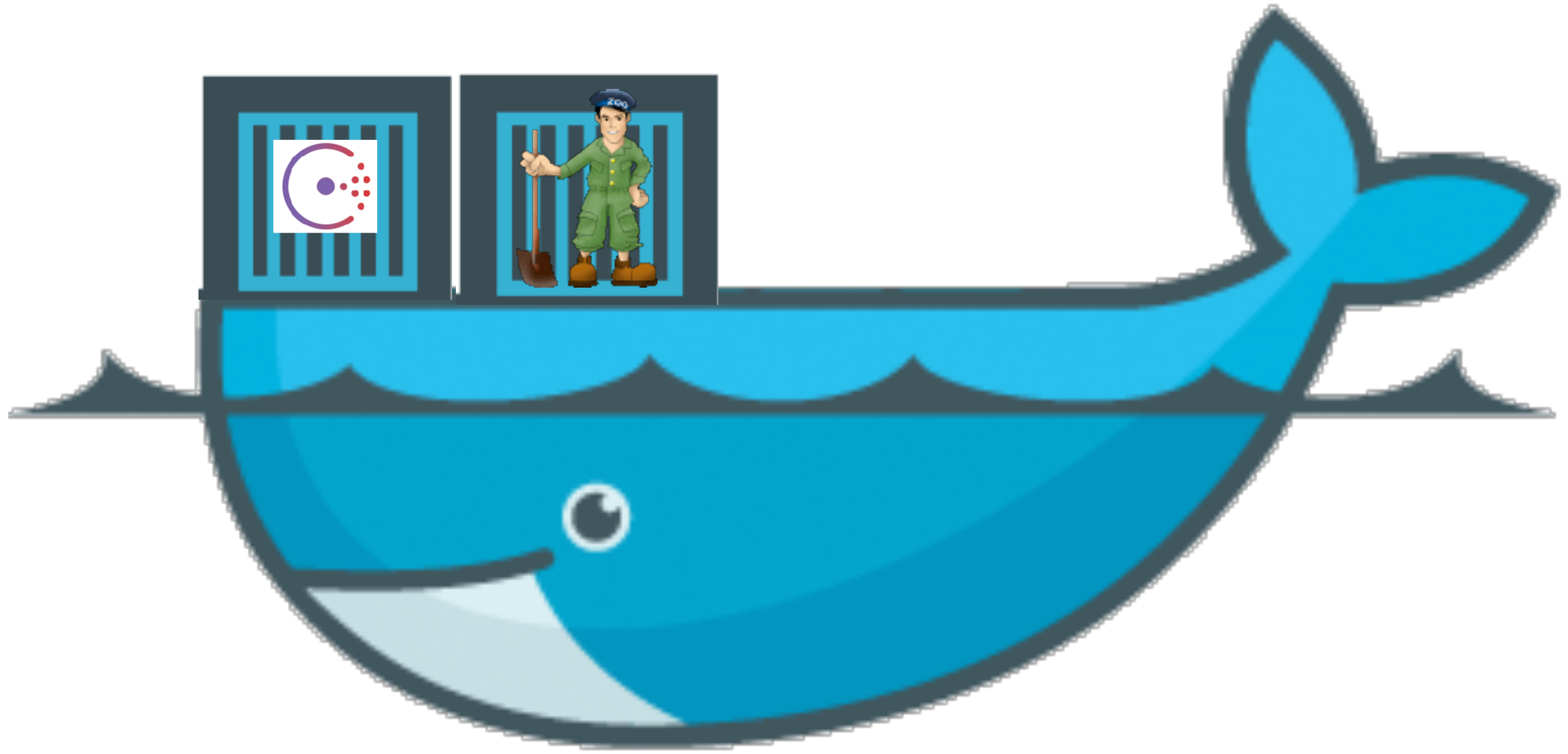


Containers

ZooKeeper

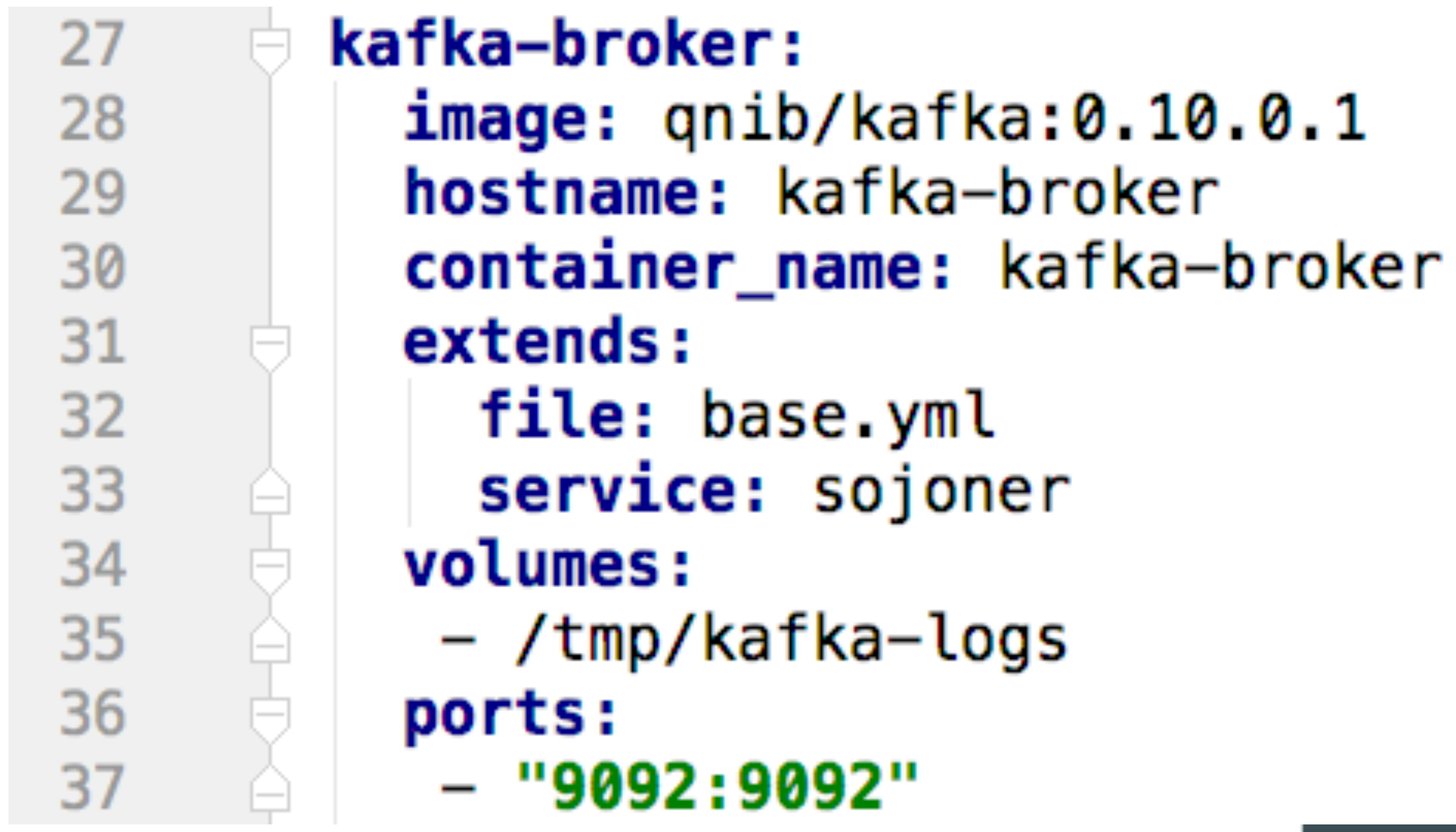


```
zookeeper:
  image: qnib/zookeeper
  hostname: zookeeper
  container_name: zookeeper
  extends:
    file: base.yml
    service: sojoner
  ports:
  - "2181:2181"
```

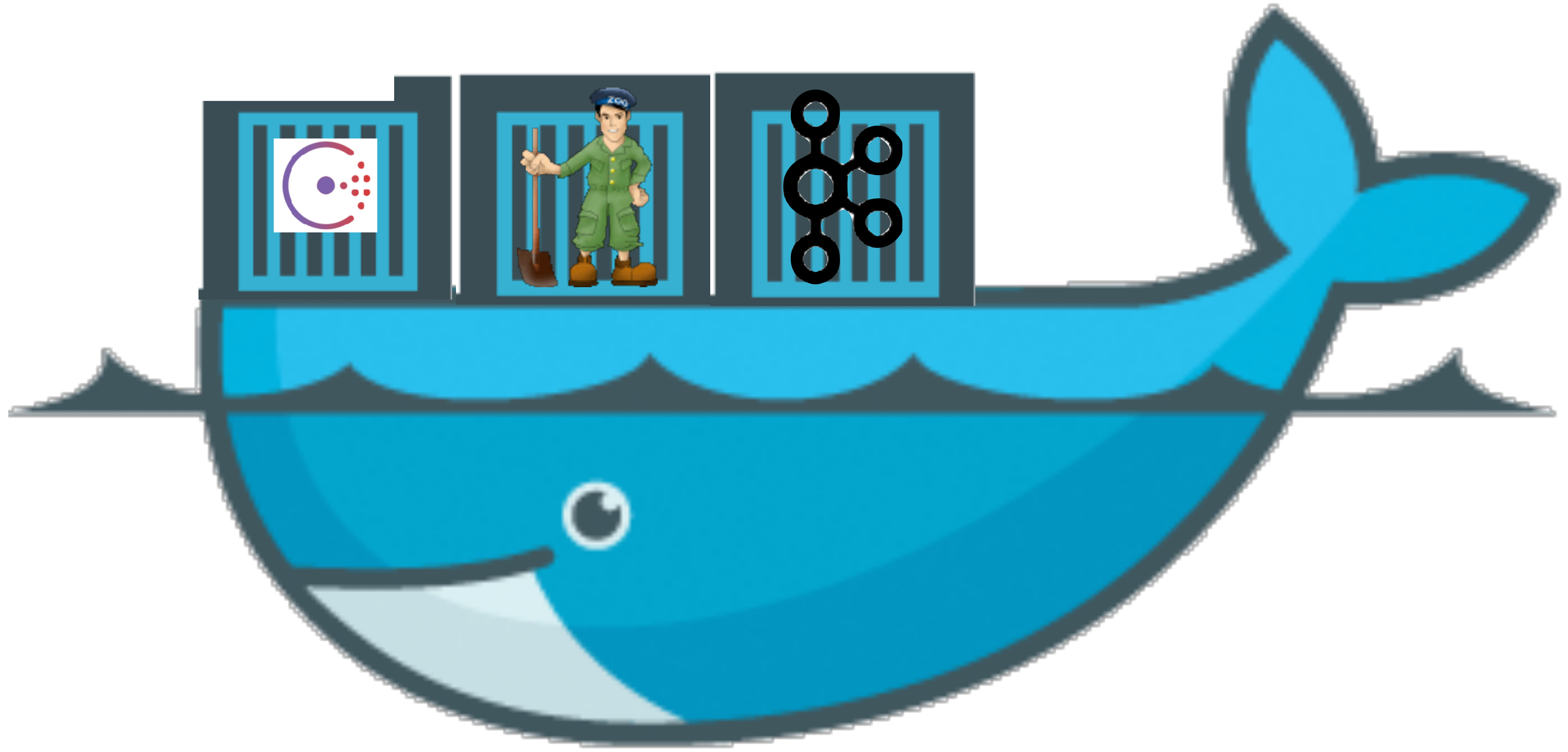


Containers

kafka

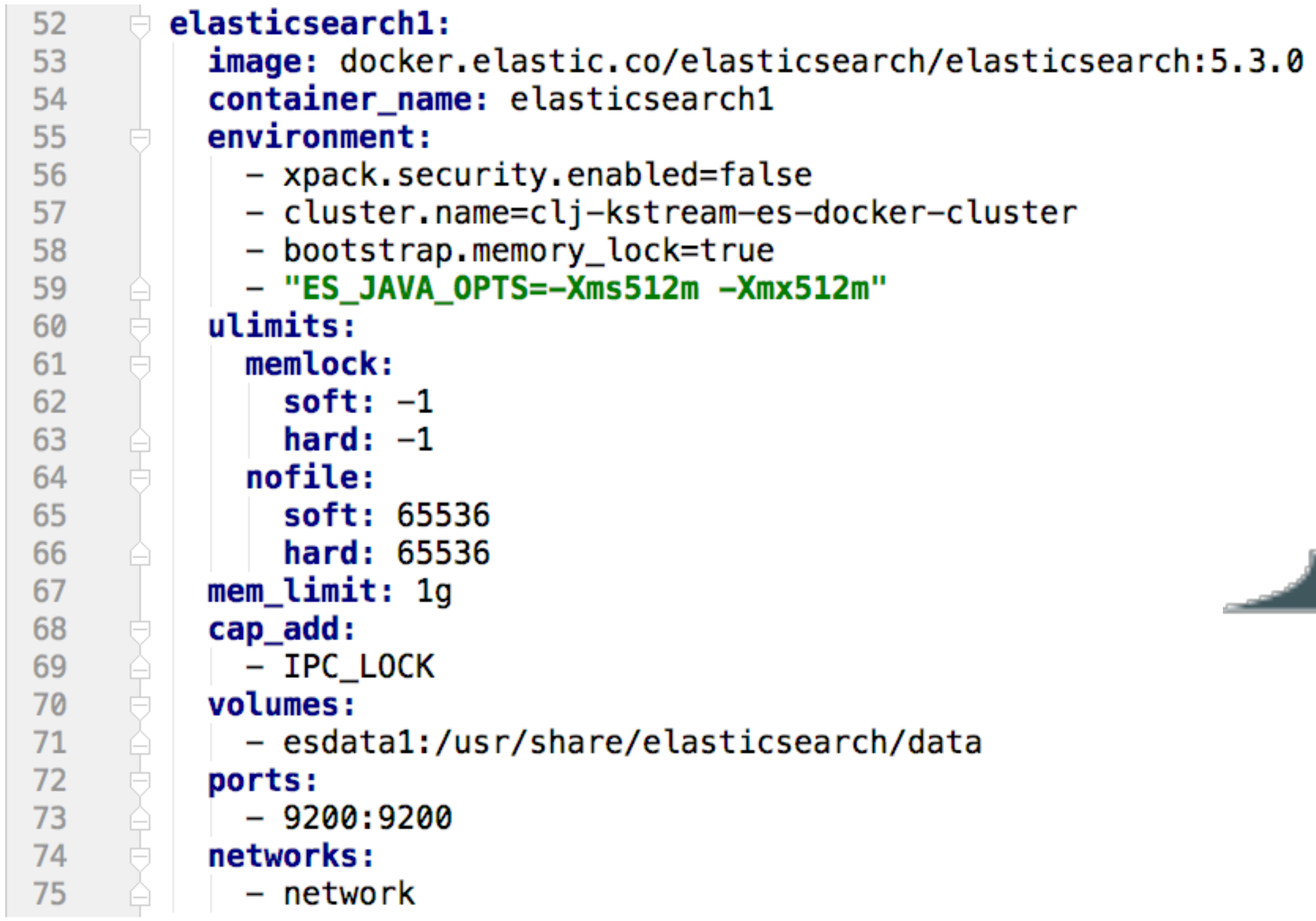


```
kafka-broker:
  image: qnib/kafka:0.10.0.1
  hostname: kafka-broker
  container_name: kafka-broker
  extends:
    file: base.yml
    service: sojoner
  volumes:
   - /tmp/kafka-logs
  ports:
   - "9092:9092"
```

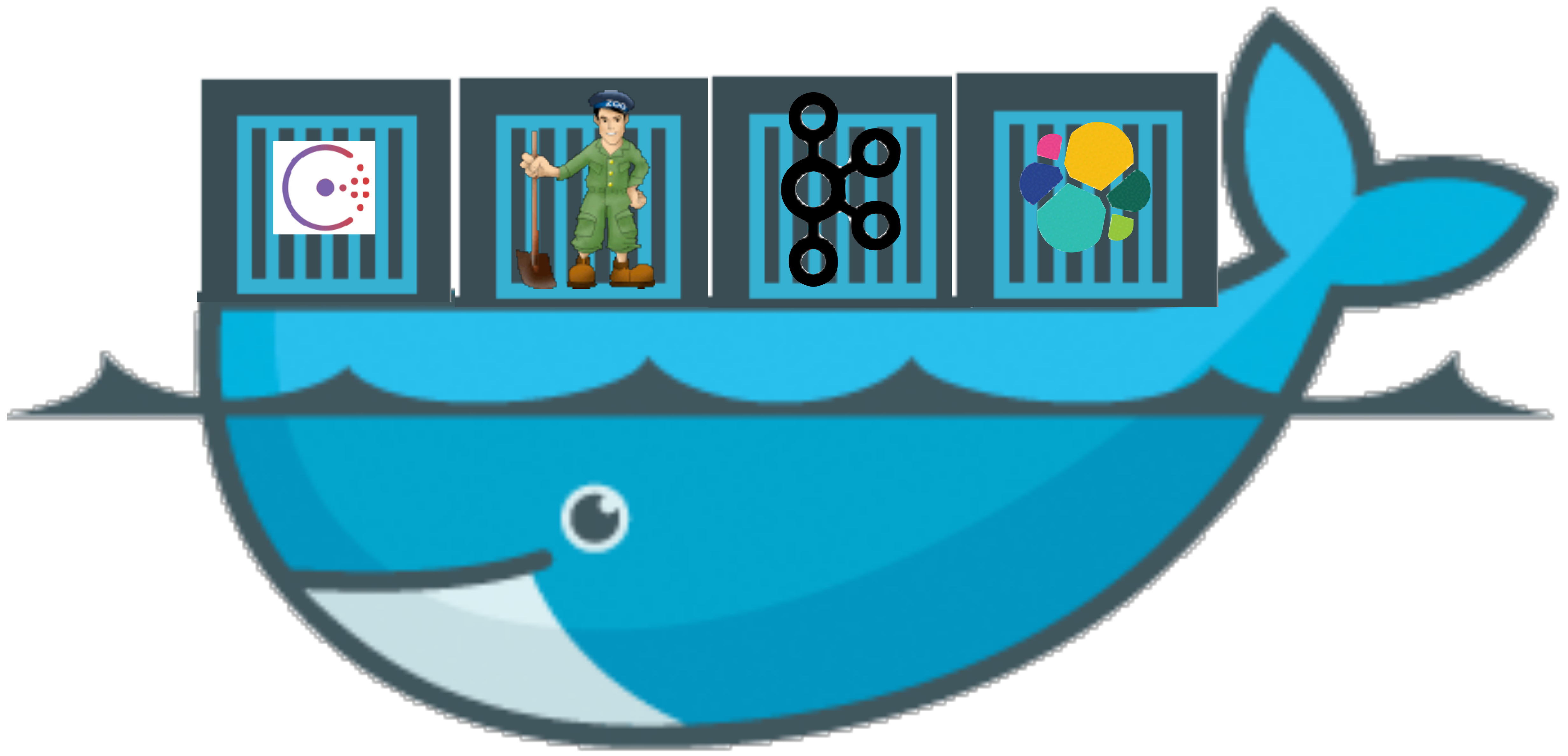


Containers

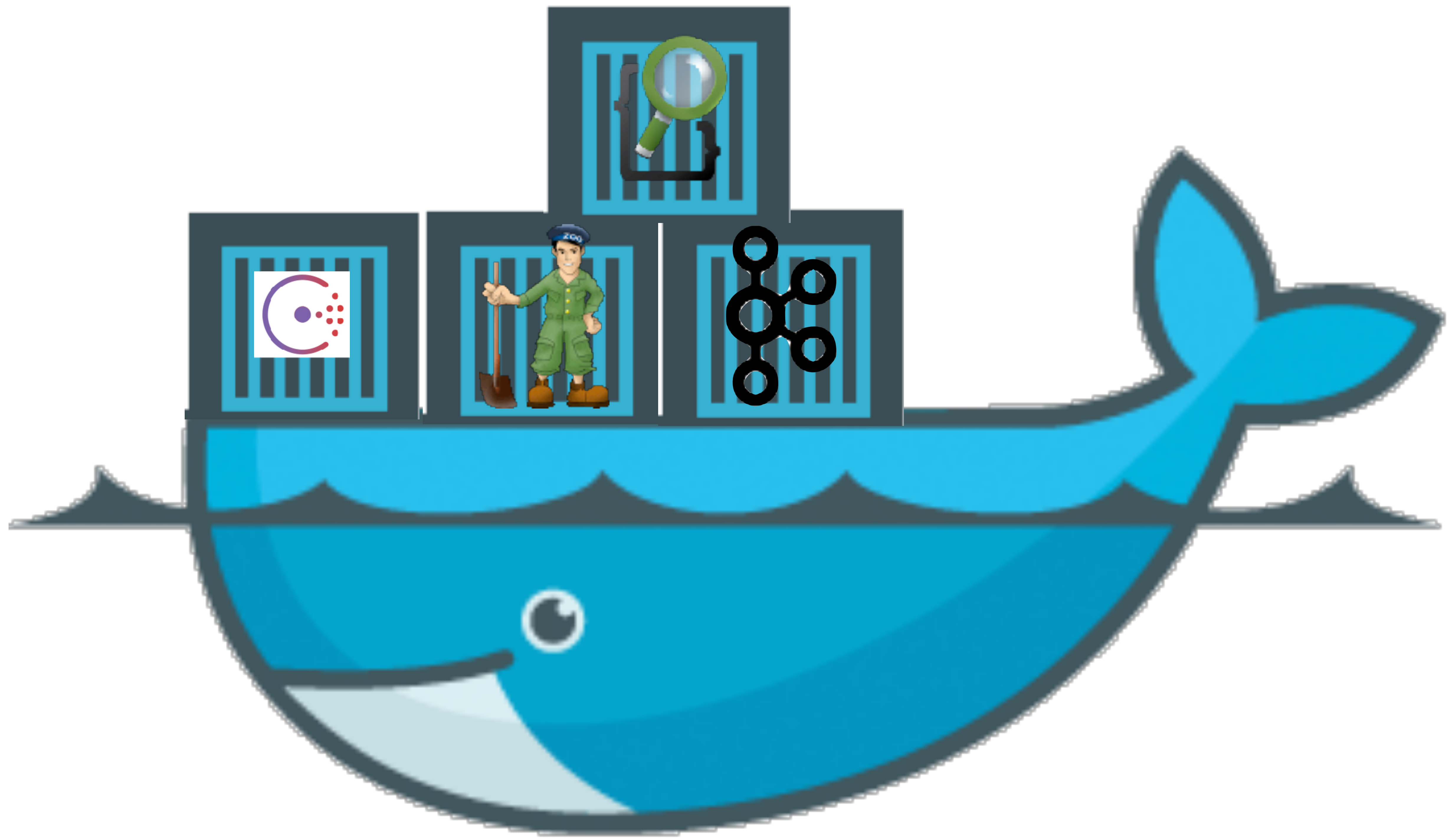
elastic



```
elasticsearch1:
  image: docker.elastic.co/elasticsearch/elasticsearch:5.3.0
  container_name: elasticsearch1
  environment:
    - xpack.security.enabled=false
    - cluster.name=clj-kstream-es-docker-cluster
    - bootstrap.memory_lock=true
    - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
  ulimits:
    memlock:
      soft: -1
      hard: -1
    nofile:
      soft: 65536
      hard: 65536
  mem_limit: 1g
  cap_add:
    - IPC_LOCK
  volumes:
    - esdata1:/usr/share/elasticsearch/data
  ports:
    - 9200:9200
  networks:
    - network
```

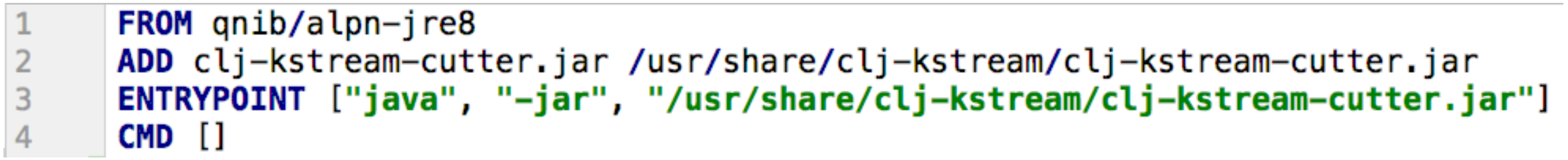


Containers

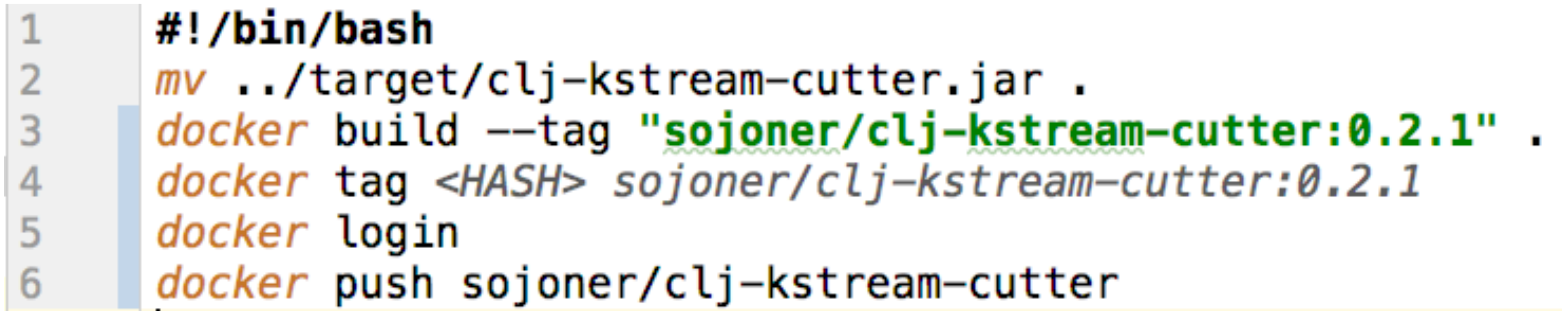


the Tools in containers

Containers



```
FROM qnib/alpn-jre8
ADD clj-kstream-cutter.jar /usr/share/clj-kstream/clj-kstream-cutter.jar
ENTRYPOINT ["java", "-jar", "/usr/share/clj-kstream/clj-kstream-cutter.jar"]
CMD []
```



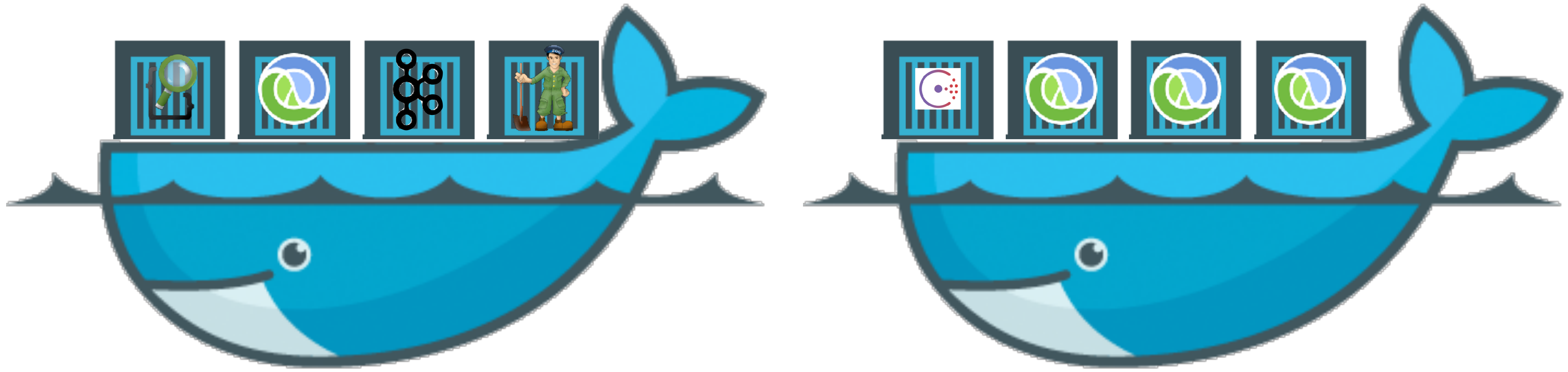
```
#!/bin/bash
mv ../target/clj-kstream-cutter.jar .
docker build --tag "sojoner/clj-kstream-cutter:0.2.1" .
docker tag <HASH> sojoner/clj-kstream-cutter:0.2.1
docker login
docker push sojoner/clj-kstream-cutter
```

Containers

```yaml
clj-kstream-cutter:
  image: sojoner/clj-kstream-cutter:0.2.0
  hostname: clj-kstream-cutter
  container_name: clj-kstream-cutter
  extends:
    file: base.yml
    service: sojoner
  command: "--broker kafka-broker:9092 --zookeeper zookeeper:2181 --input-topic logs-replay --output-topic mapped-test-json --selector msg --name stream-cut-json-field"

clj-kstream-hh:
  image: sojoner/clj-kstream-hh:0.1.0
  hostname: clj-kstream-hh
  container_name: clj-kstream-hh
  extends:
    file: base.yml
    service: sojoner
  command: "--broker kafka-broker:9092 --input-topic mapped-test-json --output-topic heavy-hitters  --window-size 1 --name stream-hh"

clj-kstream-string-long-window-aggregate:
  image: sojoner/clj-kstream-string-long-window-aggregate:0.2.2
  hostname: clj-kstream-string-long-window-aggregate
  container_name: clj-kstream-string-long-window-aggregate
  extends:
    file: base.yml
    service: sojoner
  command: "--broker kafka-broker:9092 --input-topic heavy-hitters --window-size 1 --output-topic agg-result --name stream-agg"

clj-kstream-elasticsearch-sink:
  image: sojoner/clj-kstream-elasticsearch-sink:0.0.1
  hostname: clj-elasticsearch-sink
  container_name: clj-elasticsearch-sink
  extends:
    file: base.yml
    service: sojoner
  command: "--broker kafka-broker:9092 --topic agg-result --elasticsearch http://elasticsearch1:9200 --index heavy-hitters-test-idx --index-type hh-struct"
```

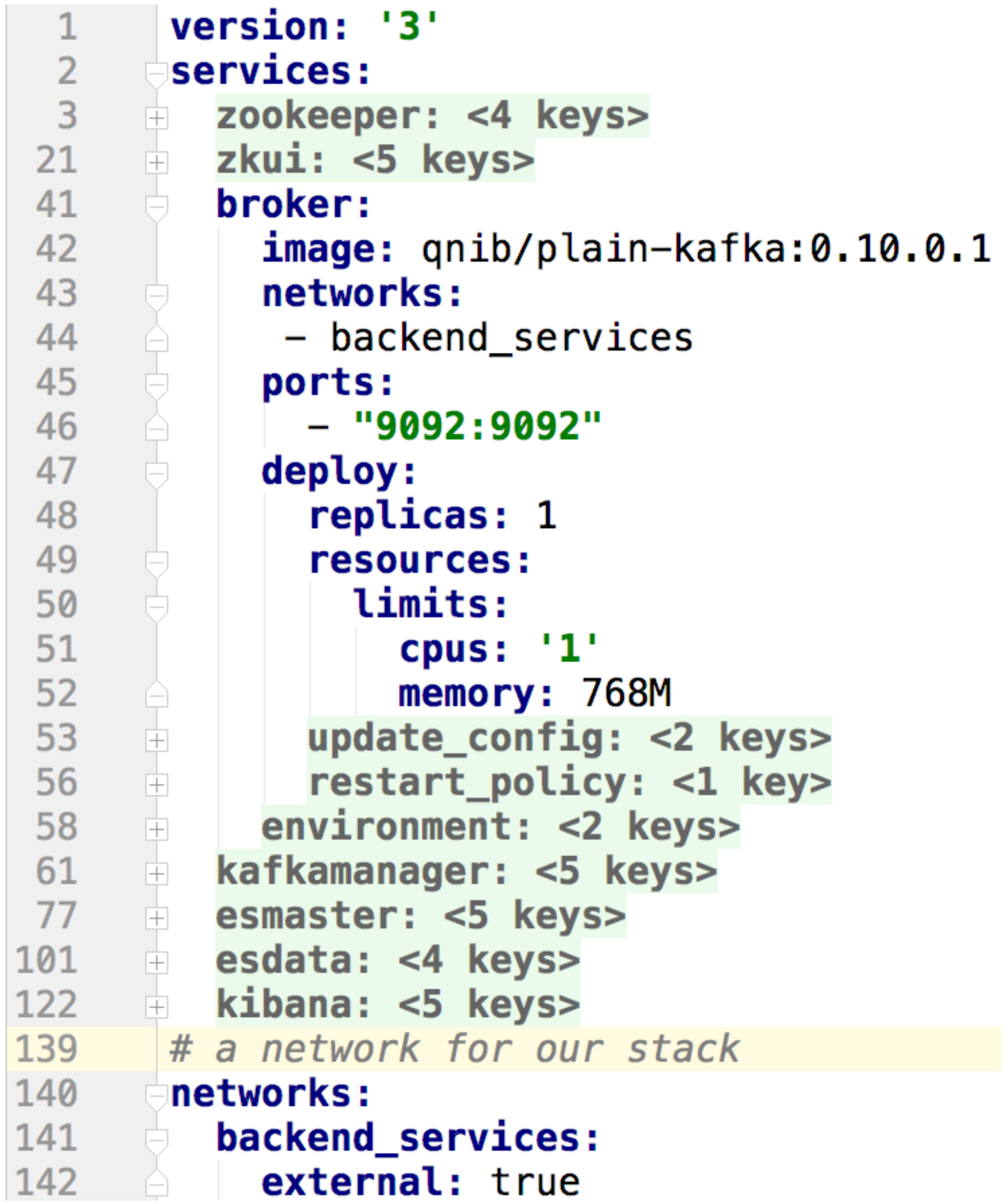
A datacenter setup

Build a Docker Swarm



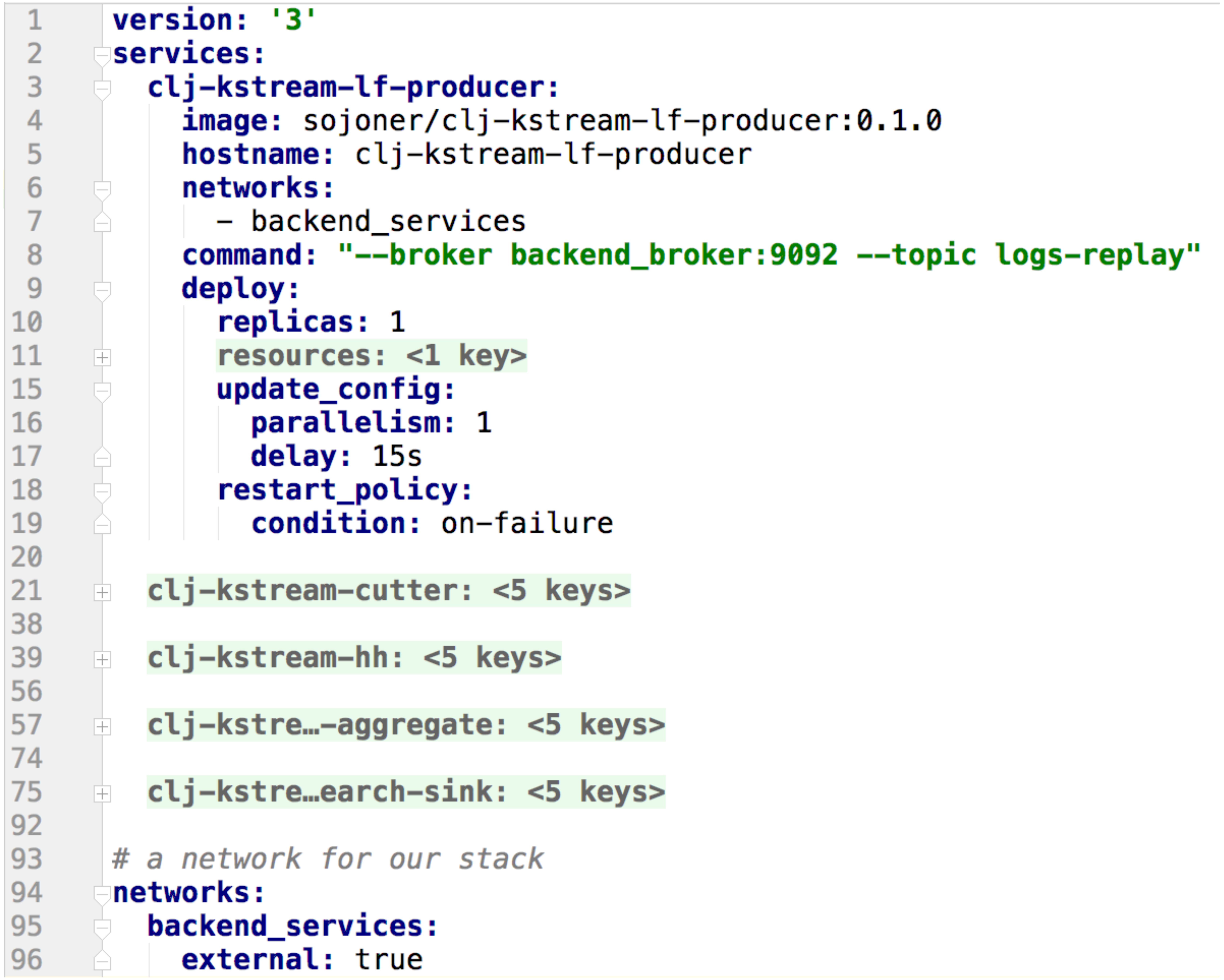
```
$ export DOCKER_HOST=tcp://my.datacenter.de:2576
```

Containers



```
version: '3'
services:
  zookeeper: <4 keys>
  zkui: <5 keys>
  broker:
    image: qnib/plain-kafka:0.10.0.1
    networks:
     - backend_services
    ports:
      - "9092:9092"
    deploy:
      replicas: 1
      resources:
        limits:
          cpus: '1'
          memory: 768M
      update_config: <2 keys>
      restart_policy: <1 key>
    environment: <2 keys>
  kafkamanager: <5 keys>
  esmaster: <5 keys>
  esdata: <4 keys>
  kibana: <5 keys>
# a network for our stack
networks:
  backend_services:
    external: true
```

Containers



```yaml
version: '3'
services:
  clj-kstream-lf-producer:
    image: sojoner/clj-kstream-lf-producer:0.1.0
    hostname: clj-kstream-lf-producer
    networks:
      - backend_services
    command: "--broker backend_broker:9092 --topic logs-replay"
    deploy:
      replicas: 1
      resources: <1 key>
      update_config:
        parallelism: 1
        delay: 15s
      restart_policy:
        condition: on-failure

  clj-kstream-cutter: <5 keys>

  clj-kstream-hh: <5 keys>

  clj-kstre…-aggregate: <5 keys>

  clj-kstre…earch-sink: <5 keys>

# a network for our stack
networks:
  backend_services:
    external: true
```

Containers

```
$ docker network create --driver overlay --attachable=true backend_services

$ docker stack deploy --compose-file backend.yml backend

$ docker stack deploy --compose-file streamprocessors.yml kstream
```

Recap

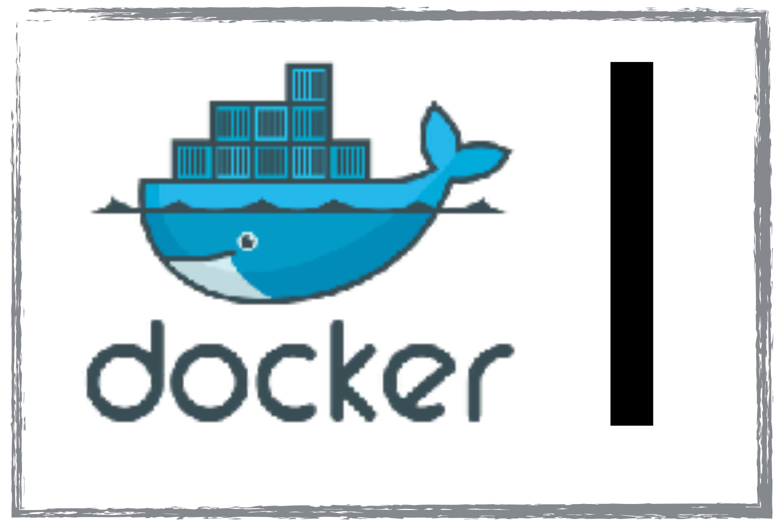
TL;DR

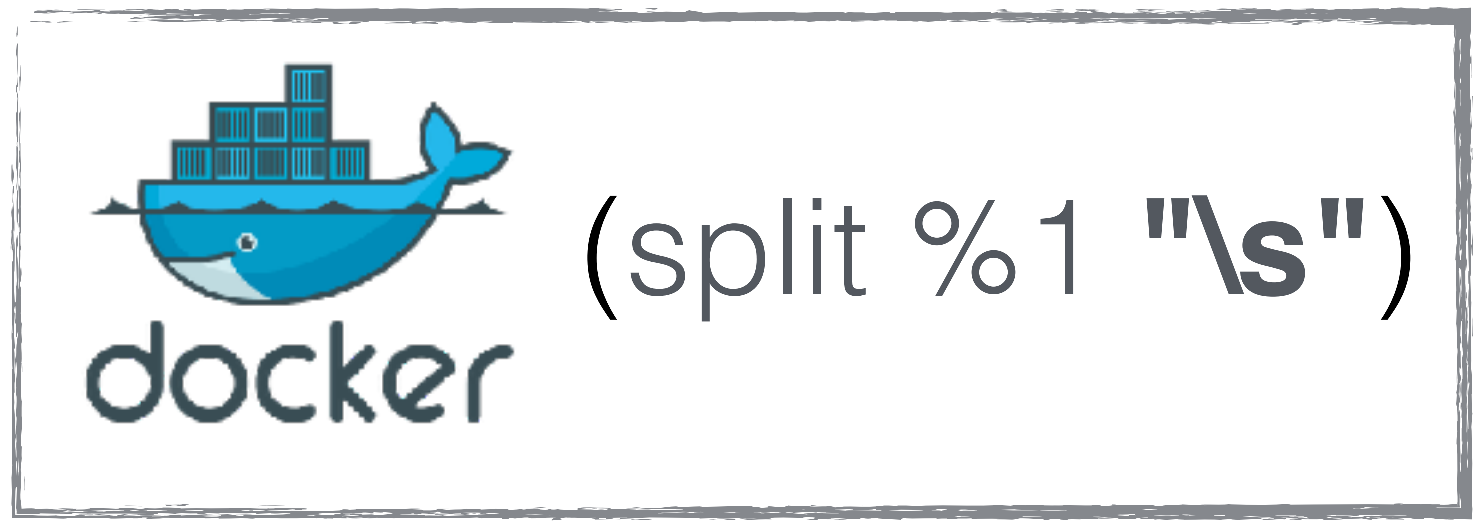
Recap

```
(split %1 "\s")
```

AS MESSAGE BROKER

AS STREAM PROCESSOR





```
(split %1 "\s")
```

GOTCHAS ?

Recap

- Kafka Streams still **at least once**
    - but **exactly ones** is coming
- **Still need capacity planning**
- **Testing / Debugging is still a challenge**
    - **Consistency** of the **state storage**
    - **Processing Time vs. Event Time**
- What about **Amdahl's law ?**
- How to manage **Docker Volumes** nicely @scale

Orientation

Recap

Human Kind:

„Take what you need“

Albert Einstein:

„Make things as simple as possible, but no simpler“

William of Ockham:

„Among competing hypotheses, the one with the fewest assumptions should be selected“

Containerizing Distributed Pipes

(thanks (listening [this]))

- http://kafka.apache.org/
- https://martin.kleppmann.com/2015/05/06/data-agility-at-strata.html
- https://www.confluent.io/blog/apache-kafka-samza-and-the-unix-philosophy-of-distributed-data/
- https://speakerdeck.com/ept/kafka-and-samza-distributed-stream-processing-in-practice
- https://github.com/mhausenblas/dnpipes
- https://en.wikipedia.org/wiki/Pipeline_%28Unix%29
- https://zookeeper.apache.org/doc/trunk/zookeeperOver.html
- https://github.com/sojoner/container-stacks/tree/master/kafkaelasticsearch
- https://kafka.apache.org/documentation/streams#streams_processor
- https://kafka.apache.org/documentation/streams#streams_dsl
- https://hub.docker.com/r/sojoner/clj-kstream-elasticsearch-sink/
- https://hub.docker.com/r/sojoner/clj-kstream-cutter/
- https://hub.docker.com/r/sojoner/clj-kstream-hh/
- https://hub.docker.com/r/sojoner/clj-kstream-string-long-window-aggregate/
- https://blog.acolyer.org/2016/07/21/time-adaptive-sketches-ada-sketches-for-summarizing-data-streams/
- https://github.com/mhausenblas/dnpipes